

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/49194>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

Chain Event Graphs: Theory and application

by

Peter Adam Thwaites

A thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Statistics

University of Warwick, Department of Statistics
April 2008

Contents:

List of Figures	iv
Abstract	vii
List of abbreviations / notation	viii
1. Introduction	
1.1 Preparatory remarks	1
1.2 Graphical Models	1
1.3 Bayesian Networks, Trees and Chain Event Graphs	6
1.4 Chain Event Graphs: Theory and application	7
2. Construction of Chain Event Graphs	
2.1 Introduction and preliminary notation	9
2.2 Trees, Events and Random Variables	9
2.3 Extending the model	14
2.4 Trees and CEGs	17
2.5 A formal definition of a Chain Event Graph	24
2.6 The construction process in practice	26
3. Reading Chain Event Graphs	
3.1 Introduction	37
3.2 Basic ideas	39
3.3 Two important results	45
3.4 Examples	49
3.5 Cuts and Chain Event Graphs	51
3.6 Semantics	57
4. Probability propagation on Chain Event Graphs	
4.1 Introduction	74
4.2 Observation of a set of positions	76
4.3 An Example	83
4.4 Lazy short-cuts	89
4.5 Local Message Passing: Observation of a set of positions	92
4.6 Propagation with general observation sets	96
4.7 An algorithm for propagation following observation of intrinsic events	107
4.8 Local Message Passing: General observation sets	113
4.9 Dynamic propagation on CEGs	124
5. Causal manipulation of Chain Event Graphs	
5.1 Introduction	127
5.2 Conditioning and manipulating	130
5.3 Defining a manipulation	134

5.4	Another look at the Machine example	
5.4.1	Our initial CEG	137
5.4.2	Conditioning on an event Λ	139
5.4.3	Manipulating to an event Λ	141
5.5	A Back Door Theorem for CEGs	
5.5.1	Background, preliminary notation and some definitions	146
5.5.2	The general form of the theorem	149
5.5.3	Singular manipulations	150
5.5.4	The theorem for Singular manipulations	152
5.5.5	A graphical condition	155
5.5.6	Examples	158
5.5.7	Blocking sets upstream of the manipulation	168
5.5.8	Using $\{\omega_X\}$ to create a blocking set	171
5.5.9	Some thoughts on general Singular manipulations	173
5.6	A Front Door Theorem for CEGs	
5.6.1	Background	174
5.6.2	A Front Door Theorem for Singular manipulations	176
5.6.3	Some further thoughts on the Front Door Theorem	181
5.7	Some final thoughts on the Causal manipulation of CEGs	182
6.	Conclusion	
6.1	Summary	184
6.2	(Causal) Estimation on CEGs	186
6.3	Equivalence classes for CEGs	188
6.4	Further work	191
References		193

List of Figures:

1. Introduction

Figure 1: Tree for tossing two fair coins	3
---	---

2. Construction of Chain Event Graphs

Figure 1: Tree for motivating example	10
Figure 2: A Probability Tree with some asymmetry	14
Figure 3: Probability Tree for Example 2	19
Figure 4: Tree for Blood Example	28
Figure 5: Tree for Blood Example showing patient groups and equivalent probabilities	29
Figure 6: Augmented Tree for Blood Example	30
Figure 7: Blood Example graph following Step 2 of the construction process	32
Figure 8: Final CEG for Blood Example	33
Figure 9: Tree for Machine Example	34
Figure 10: Augmented tree for Machine Example	35
Figure 11: Machine Example graph following Step 2 of the construction process	36
Figure 12: Final CEG for Machine Example	36

3. Reading Chain Event Graphs

Figure 1: Tree for Example 1	41
Figure 2: Example 1 Tree with probabilities added	42
Figure 3: CEG for Example 1	43
Figure 4: CEG for Blood Example	49
Figure 5: CEG for Machine Example	51
Figure 6: BN and CEG for Example 5	52
Figure 7: BN and CEG for Example 6	53
Figure 8: CEG for Example 7	55
Figure 9: CEG for Example 8	56
Figure 10: Abbreviated CEG for Examples 9, 10 and 11	58

4. Probability propagation on Chain Event Graphs

Figure 1: CEG for Blood condition Example	84
Figure 2: CEG with pre-observation probabilities	84
Figure 3: Graph following Algorithm step (1)	85
Figure 4: Graph following Algorithm steps (2) and (3)	85
Figure 5: Graph following Algorithm step (4)	86
Figure 6: Graph following first application of Algorithm step (5)	86
Figure 7: Graph following final application of Algorithm step (5)	87
Figure 8: CEG following completion of our algorithm	88
Figure 9: Final CEG	88
Figure 10: Abbreviated CEG	90
Figure 11: CEG for Example 2	98

Figure 12: CEG conditioned on the event Λ_1	99
Figure 13: Tree showing the event Λ_2	99
Figure 14: CEG conditioned on the event Λ_2	100
Figure 15: Subgraph of C defined by the event Λ_1	101
Figure 16: Subgraph of C defined by the event Λ_2	101
Figure 17: CEG showing component paths of Λ	109
Figure 18: The subgraph $G_\Lambda(C)$	109
Figure 19: Graph following Algorithm step (3)	111
Figure 20: CEG following completion of our algorithm	111
Figure 21: Conditioned CEG C_Λ	111
Figure 22: CEG for Example 4	114
Figure 23: CEG C for our example	120
Figure 24: CEG C with event Λ indicated	120
Figure 25: $G_\Lambda(C)$ for the event Λ	121
Figure 26: $G_\Lambda(C)$ with potentials and emphases added	121
Figure 27: Updated CEG (with unchanged position labels)	122
Figure 28: Tree T with $T_{(\Lambda)}$ outlined in green	123
Figure 29: A Dynamic Bayesian Network	124

5. Causal manipulation of Chain Event Graphs

Figure 1: Bayesian Network with three binary variables	130
Figure 2: CEG corresponding to BN in Figure 1	131
Figure 3: $G_\Lambda(C)$ for $\Lambda = b_0$	131
Figure 4: $G_\Lambda(C)$ with edge-labels	132
Figure 5: C_Λ for $\Lambda = b_0$	133
Figure 6: $C_{do\Lambda}$ for $\Lambda = b_0$	134
Figure 7: $G_\Lambda(C)$ for the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_5$	135
Figure 8: $C_{do\Lambda}$ for the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_5$	135
Figure 9: CEG for Machine Example	139
Figure 10: CEG showing probabilities	139
Figure 11: Conditioned CEG C_Λ	140
Figure 12: Manipulated CEG $C_{do\Lambda}$ for option (a)	142
Figure 13: Manipulated CEG $C_{do\Lambda}$ for option (b)	143
Figure 14: Manipulated CEG $C_{do\Lambda}$ for option (c)	144
Figure 15: Alternative CEG C	145
Figure 16: Alternative CEG $C_{do\Lambda}$	145
Figure 17: BN for Example 1	146
Figure 18: CEG fragment	158
Figure 19: Bayesian Network and CEG for Example 2	158
Figure 20: Derived Graph $D_{do\Lambda_x}$ for Example 2	159
Figure 21: CEG for Example 3	161
Figure 22: Derived Graph $D_{do\Lambda_x}$ for Example 3	163
Figure 23: Bayesian Network and CEG for Example 5	178
Figure 24: Derived Graph $D_{do\Lambda_x}$ for Example 5	178

6. Conclusion

Figure 1: Three simple DAGs	188
Figure 2: Six DAG-structures	189

Acknowledgements

I would like to acknowledge the help and support of the staff of the University of Warwick Statistics Department; in particular Paula Matthews and my supervisor Professor Jim Smith, whose enthusiasm for his subject and hence my work, has kept me buoyant through some fairly rough weather.

I would also like to acknowledge the interest shown in my work by Milan Studeny (Prague) and Robert Cowell (London).

Declaration

I declare that this thesis is my own work, and has not been submitted for examination elsewhere. Some material has been adapted from papers published jointly with Professor Smith, listed in the references. Where this is the case what appears in the text is based on my own contribution to these papers, and the papers are cited at the appropriate point in the text.

Abstract:

This thesis is concerned with the Graphical model known as the Chain Event Graph (CEG) [1][60][61], and develops the theory that appears in the currently published papers on this work. Results derived are analogous to those produced for Bayesian Networks (BNs), and I show that for asymmetric problems the CEG is generally superior to the BN both as a representation of the problem and as an analytical tool.

The CEG is designed to embody the conditional independence structure of problems whose state spaces are asymmetric and do not admit a natural Product Space structure. In this they differ from BNs and other structures with variable-based topologies. Chapter 1 details researchers' attempts to adapt BNs to model such problems, and outlines the advantages CEGs have over these adaptations. Chapter 2 describes the construction of CEGs.

In chapter 3 I create a semantic structure for the reading of CEGs, and derive results expressible in the form of context-specific conditional independence statements, that allow us to delve much more deeply into the independence structure of a problem than we can do with BNs. In chapter 4 I develop algorithms for the updating of a CEG following observation of an event, analogous to the Local Message Passing algorithms used with BNs. These are more efficient than the BN-based algorithms when used with asymmetric problems.

Chapter 5 develops the theory of Causal manipulation of CEGs, and introduces the singular manipulation, a class of interventions containing the set of interventions possible with BNs. I produce Back Door and Front Door Theorems analogous to those of Pearl [42], but more flexible as they allow asymmetric manipulations of asymmetric problems. The ideas and results of chapters 2 to 5 are summarised in chapter 6.

List of abbreviations / notation

The following is a list of the abbreviations and the principal symbols used in this thesis.

Abbreviations

BN	Bayesian Network
CBN	Causal Bayesian Network
CEG	Chain Event Graph
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
ET	Event Tree

Trees and CEGs – topology

$T(V(T), E(T))$	A Probability Tree
$V(T) = \{v\}$	The vertex set of Probability Tree T
v	A vertex of T , with possible subscript eg. v_i
$E(T) = \{e\}$	The edge set of Probability Tree T
V^I	The set of intermediate vertices of T – Definition 2
$e(v, v')$	(An) edge joining vertices v and v'
$T^A(V, E_d, E_u)$	An Augmented Tree – Definition 7
$T_\Lambda(V, E)$	A subtree of T defined in Definition 27
$T_{(\Lambda)}$	A subtree of T described in section 4.8
$C(W(C), E_d(C), E_u(C))$	A Chain Event Graph – Definitions 9 and 13
$W(C) = \{w\}$	The position (vertex) set of Chain Event Graph C
w	A position of C , with possible subscript eg. w_i
$E_d(C) = \{e\}$	The directed edge set of Chain Event Graph C
e	A directed edge of C (also edge of T – see above), with possible subscript eg. e_j
$E_u(C)$	The undirected edge set of Chain Event Graph C
w_0, w_∞	The root and sink nodes of C
u	A stage of C – Definition 12
$w(0, *, 1)$	Shorthand labelling of positions introduced in ch 3 Example 5
$w_1 \sim w_2$	Adjacent positions
$w_1 \prec w_2$	Position w_1 precedes position w_2 on some path
$e(w_1, w_2)$	A (directed) edge between w_1 and w_2
$\mu(w_1, w_2)$	A path-segment or subpath between w_1 and w_2
$G_\Lambda(C)(V, E)$	A subgraph of C defined in Definition 25
$D_\Lambda(V, E)$	A subgraph of C defined in Definition 29
$D_{do\Lambda}(V, E)$	A subgraph of C defined in Definition 30
C_Λ	The CEG derived from T_Λ – Definition 28
$C_{do\Lambda}$	The CEG derived from $D_{do\Lambda}$ – Definition 31

Random Variables

A, B, C	Measurement variables used with BNs, or criterion variables used with Trees / CEGs, with possible subscripts
X_i, Y_i	Variables defined on Probability Trees / CEGs – Definitions 4, 5, 10 and 11

$X_i \equiv X_j$	Variables defined on Probability Trees / CEGs, which are stage-equivalent – Definitions 6 and 12
$Y_i \equiv Y_j$	Variables defined on Probability Trees, which are equivalent – Definition 8
$X(u), Y(w), Z(u), Z(w)$	Variables defined on CEGs – Definitions 14, 15, 16 and 17
$X(u_m) \equiv X(u_n)$	Variables defined on CEGs, which are equivalent – Definition 19
$Y(w_k) \equiv Y(w_l)$	Variables defined on CEGs, which are equivalent – Definition 20
Z	A set of variables used with BNs – a Back Door blocking set
$Q(X)$	A set of variables used with BNs – the parents of the variables X

Events

ϕ	The empty set (also used for empty set of variables)
λ	An atomic root-to-sink (leaf) path in C (T), with possible subscript eg. λ_i
Λ	An event – a union of atomic root-to-sink paths in C , with possible subscript eg. Λ_j
$\Lambda(v)$	The union of all root-to-leaf paths in T passing through v – Definition 3
$\Lambda(e), e \in E_d(C), E(T)$	The union of all root-to-sink (leaf) paths in C (T) passing through the edge e – Definitions 3 and 13
$\Lambda(w)$	The union of all root-to-sink paths in C passing through the position w – Definition 13
$\Lambda(w_1, w_2)$ $= \Lambda(w_1) \cap \Lambda(w_2)$	The union of all root-to-sink paths in C passing through the positions w_1 and w_2 – Definition 22
$\Lambda(e(w_1, w_2))$	The union of all root-to-sink paths in C utilising the edge $e(w_1, w_2)$
$\Lambda(u)$	The union of all root-to-sink paths in C passing through a position w such that $w \in u$ – Definition 18
$\Lambda_x, \Lambda_y, \Lambda_z$	Events used with Singular manipulations – see sections 5.5.3 on
$\Lambda(\mu(w_1, w_2))$	The union of all root-to-sink paths in C utilising the path-segment $\mu(w_1, w_2)$ – Definition 21
a, b, c	Outcomes associated with measurement or criterion variables – the unions of sets of $\{\lambda\}$
$a_i b_j c_k$	An event – the union of a set of $\{\lambda\}$ – Definition 1
do	The manipulation of a system to a particular event, eg. $do \Lambda$

Probabilities

π	A probability
$\pi(w_2 \mid w_1)$ $= \pi(\Lambda(w_2) \mid \Lambda(w_1))$	The probability of reaching position w_2 given that we have reached position w_1
$\pi_e(w_2 \mid w_1)$ $= \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1))$	The edge-probability for edge $e(w_1, w_2)$

$\pi_\mu(w_2 \mid w_1)$	The probability of utilising path-segment $\mu(w_1, w_2)$
$= \pi(\Lambda(\mu(w_1, w_2) \mid \Lambda(w_1))$	given that we have reached position w_1
$\hat{\pi}$	A probability on a conditioned or manipulated CEG
π_Λ	A probability on D_Λ
$\pi_{do\Lambda}$	A probability on $D_{do\Lambda}$
θ_i	An edge-probability on a CEG
ϕ_i	An edge-probability on a conditioned or manipulated CEG
$\Phi(w)$	The emphasis of w – used in Local Message Passing algorithms in sections 4.5 and 4.8
$\tau_e(w' \mid w)$	The potential of $e(w, w')$ – used in Local Message Passing algorithms in sections 4.5 and 4.8

1. Introduction

1.1 Preparatory remarks

This thesis is concerned with the type of graphical model known as the Chain Event Graph (CEG), introduced by J.Q. Smith in [45][53], and featuring in [61][60][62]. The CEG is a graph designed to embody the conditional independence structures of problems whose state spaces are asymmetric and do not admit a natural Product Space structure. In this they differ significantly from Bayesian Networks (BNs) and other structures with variable-based topologies. Chapters 2 and 3 describe the construction of CEGs, and techniques for reading the conditional independence structure of a model from them. Both [45][53] contain material on these aspects, but the work presented here is completely new, and I approach the subject matter in a radically different way to the approach used in these two papers. Chapters 4 and 5 contain major new developments in the areas of propagation on, and causal analysis of CEGs. In chapter 4 I provide easy-to-use algorithms for CEGs that are directly analogous to Local Message Passing and Junction Tree propagation algorithms used on BNs [50][33][56]. Chapter 5 looks at how we can represent and analyse causal manipulations using a CEG, and contains new results analogous to Pearl's Back Door and Front Door theorems [40][41] for BNs. As with chapters 2 and 3, the content of these chapters is completely new, and my approach differs significantly from that used by Smith in [45][53]. Chapter 6 contains outlines of where I would hope to take this research in the future, as well as sketches of ideas that have been as yet insufficiently developed so as to appear in the main body of the thesis.

The CEG is a very recent addition to the class of graphical models, and so the rest of chapter 1 is devoted to a (necessarily personal) description of their genesis and purpose.

1.2 Graphical models

When a statistician talks about a graphical model, they are generally referring to a model based on a graph or network consisting of vertices or nodes connected by edges or arcs (which may be directed or undirected), where the vertices are in some way associated with random variables and the edges indicate some sort of dependency between these variables. To complete the model, some probability distribution is imposed on the graph which satisfies the conditional independence structure implied by the graph's missing edges. It is this type of model which Lauritzen describes in his seminal work *Graphical models* [29].

An important subset of these models is the set of Bayesian Networks. Jensen [25] states that a BN consists of a set of variables and a set of directed edges between variables forming a directed acyclic graph (DAG), together with a set of conditional probability tables $P(A \mid B_1, \dots, B_n)$ for each variable A with parents B_1, \dots, B_n . The BN is thus a means of describing models wherein counts or probabilities are assigned to cells corresponding to values of a vector of variables. Missing edges between vertices indicate a lack of direct association between variables, and directed edges between vertices indicate some sort of (usually temporal) ordering.

The Bayesian Network appeared out of the mass of other graphical models in the mid 1980s. Wermuth and Lauritzen used DAGs in [64], Pearl used the name *Bayesian Network* in [38], and introduced BNs to expert systems in [39]. They have since become a highly successful graphical structure, principally because one can use the graph for problem representation; problem interrogation (using the d-separation theorem to establish the conditional independence structure of the problem); probability calculation; and as a framework for embedding uncontrolled models into controlled ones. Their continuing success can largely be attributed to this combination of simple model description and comparatively straightforward model analysis.

Unsurprisingly, researchers have wished to extend the scope of the BN, and this led rapidly to an attempt to interpret the directionality of the edges as in some way *causal*. These users needed to manipulate the topology of the BN so as to model *causal interventions*, and from this developed the theory of Causal Bayesian Networks (CBNs) in the works of Pearl [40][41][42], Dawid [12] and Lauritzen [30], among others. Using these graphs the effects of manipulations of the system (usually setting a specific variable to a specific value) can be analysed.

The development of BNs has involved practitioners from many disciplines, and this has led to mutations and hybrid forms. In Decision Analysis BNs exist in the form of Influence Diagrams, which contain decision nodes and utility function nodes as well as the nodes representing random variables (see for example [51], [37] and also [52] where Smith equates BNs and Influence Diagrams in a statistical modelling context rather than a Decision Analysis one).

Given the comparative ease with which the BN can be adapted for use in different areas, the rapid uptake in its use by researchers in many areas is hardly surprising. Probably the most avid users and developers of the theory are those within the AI community.

As already noted, the title *Graphical Models* is usually intended to refer to the types of

models described above or their close relatives. We could, however, equally well apply the title to any graph or network onto which has been imposed a probability model. The simple Tree in Figure 1 is thus a graphical model, and completely describes a model wherein two *fair* coins are tossed.

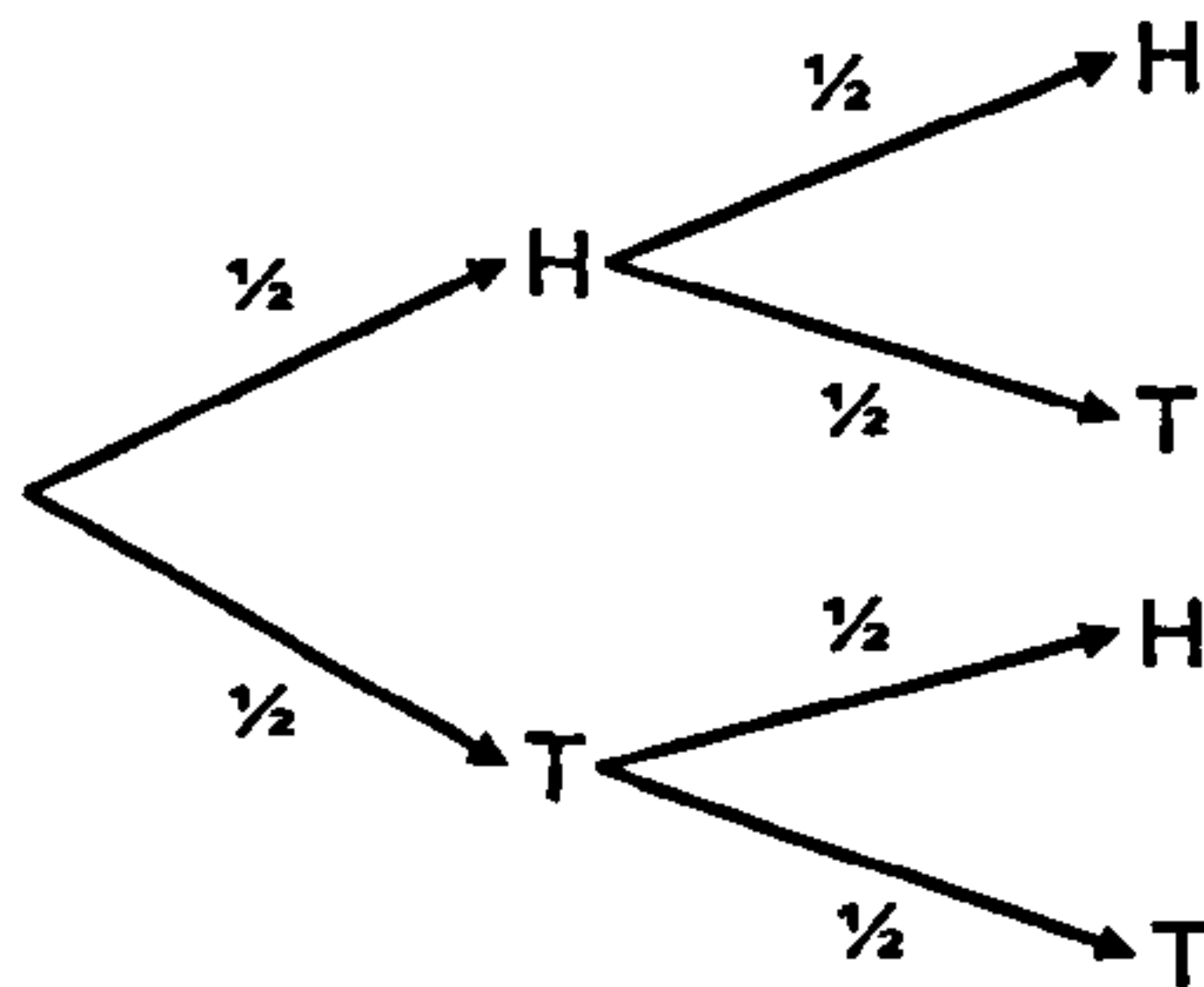


Figure 1: Tree for tossing two fair coins

Shafer in [49] reminds us that a *probability tree ... includes a sample space, but adds further structure to it*, so here the vertices represent states in the unfolding of the process; the first H means that the first coin tossed has landed Heads, and the next H on this branch means that both coins have landed Heads. The edges carry the information about how the process unfolds given the current state, so the edge connecting the root node to the first H indicates the landing of the first coin as Heads, and the edges connecting the first H and the first T to the second H on their respective branches indicate the landing of the second coin as Heads, given that the first coin landed Heads or Tails. The edges carry conditional probabilities, such as the probability that the second coin is Heads given that the first coin was Tails. Such trees have been used certainly since the 17th Century; Shafer in [49] reproduces a tree drawn by Christiaan Huygens in 1676 [22]. Like BNs, Trees have been used extensively in Decision Analysis (see for example [51]), and also less extensively in the study of Causality, where Shafer [49] treads a fairly lonely path.

Both BNs and Trees have proved useful in the representation and analysis of statistical models, but both have shortcomings. Trees allow an investigator to describe exactly the unfolding of a process, but are inevitably rather bulky which renders their use for large problems unrealistic. The major problem with using Trees for rigorous modelling and analysis is that one cannot read (conditional) independence properties from a Tree, and this has led to their eclipse by BNs as graphical representations of discrete statistical models.

BNs have proved very good graphical representations for many discrete joint probability distributions, but those distributions that are most successfully represented by a BN tend

to have a high degree of symmetry. Dependencies in many Bayesian statistical models (such as hierarchical models) exhibit an amenable type of symmetry. The state space for a BN is a Product Space wherein each atom corresponds to a unique vector of values of the problem variables. BNs have proved less successful at representing models with non-symmetric sample space structures. French and Insua in [15], writing about Decision Trees and Influence diagrams, state that:

Decision Trees have a difficulty in that for many problems they rapidly become very large ... Influence diagrams are a much more compact representation. However their advantage in this respect is, in a sense, illusory. Decision trees can represent asymmetric problems, ie. problems in which a particular choice of action at a decision node makes available different choices of action at subsequent decision nodes than those available after an alternative choice. Such asymmetric problems are the rule rather than the exception in decision analysis.

For Decision trees and Influence diagrams read Trees and BNs, and for asymmetric problems in decision analysis read asymmetric processes in areas such as biological regulation [7], risk analysis [2] and Bayesian policy analysis [14].

One of the running examples in this thesis, introduced formally in chapter 2, concerns the treatment and effects of a particular genetic blood condition. In this example, if a patient does not have the blood condition we do not record whether they are Rhesus +ve or -ve, nor a *quality of life* score; if a patient does have the condition and has blood group O, we record a *quality of life* score, but not whether they are Rhesus +ve or -ve; whereas if they have the condition and blood group B we record both these facets. Clearly the state space for this problem cannot be expressed as a Product Space. Although we can create variables which describe the problem, not all atoms will correspond to a vector of values of each variable. For some values of our *root* variable we know nothing about the values of other variables in the problem. To model this using a BN we would need to have variables taking a value corresponding to *nothing happens*, and in any contingency table drawn from a BN-representation of the problem we would have systematically missing information.

Attempts have been made to combine BNs and Trees so as to benefit from the strengths of both. Probably the most significant development in this area is the context-specific Bayes Net [3], where tree-structured conditional probability tables are annexed to the vertices of a BN to allow for the analysis of context-specific independence properties (in-

dependence properties that only exist given certain values of specific variables). Friedman and Goldszmidt in [16] use a very similar idea when looking at the process of Learning a BN. They suggest that this is best done by learning *local structure* via looking at the conditional probability distributions that describe the BN. From analysing context-specific independencies at each vertex they produce Decision Trees and *default tables*, which contain fewer rows than the original conditional probability distribution table, for that vertex. The context-specific Bayes Net is not however a universal panacea – the afore-mentioned blood condition example could not be expressed as a context-specific Bayes Net; and in fact any process (such as a treatment regime) whose unfolding depends on the state of the system at any particular point and the values of specific covariates at that point, cannot be efficiently expressed as context-specific Bayes Nets, although they can always be expressed efficiently as Trees.

The context-specific BN is not the only hybrid structure to have been developed. Covaliu and Oliver in [9] use Decision Trees to augment Influence diagrams with extra information, edges carrying labels about which values of variables imply dependence between two nodes. They call the resultant graphs Sequential Decision Diagrams. The Chain Graph [34][59] is a mixed graph containing both undirected and directed edges between variables. Without getting too involved in the subtleties of the Chain Graph, we can say that an undirected edge between two vertices suggests some sort of dependence which is symmetric – there is no temporal or causal ordering, but simply an association. Andersson, Madigan and Perlman in [1] have now extended the idea of the Chain Graph to represent different classes of Markov-equivalent graphs.

Other graphical representations have been proposed in response to problems encountered in using existing representations to perform particular tasks. Since a BN is an inefficient representation of an asymmetric problem, propagation of information in such a problem using a BN is comparatively slow. Jaeger’s Probabilistic Decision Graph [23][24] uses a Tree-like structure to allow fast propagation of information, but cannot express all the conditional independencies of a BN.

Causal analysis on BNs can also become problematic even with highly symmetric problems – many manipulations are simply not expressible as the setting of a specific variable to a specific value. Furthermore the nature of a manipulation is often dependent on the state of a system and the values of covariates at a specified time. In [41] section 4.2, Pearl briefly touches on this idea. He notes that interventions may involve policies whereby a variable X responds to some other variable(s) Z through either a functional relationship $x = g(z)$,

or a stochastic relationship whereby X is set to x with some probability dependent on the value(s) z . If we follow causal analysis convention and describe an unmanipulated system as *idle*, we can see that even a symmetric idle system can give rise to a highly asymmetric structure given a particular manipulation rule or policy.

1.3 Bayesian Networks, Trees and Chain Event Graphs

It is clear that there is a vital need for a simple graph which, without the need for further modification, will allow us to represent and analyse discrete asymmetric processes. That the graph must be less bulky than a Tree for large problems is a prerequisite. If the use of this graph by researchers and data analysts is to become widespread, we really need to be able to replicate the principal results established for BNs. In particular we need to be able to use the graph for problem representation; problem interrogation (using an analogue of the d-separation theorem to establish the conditional independence structure of the problem); probability calculation; and as a framework for embedding uncontrolled models into controlled ones. We will also want to be able to use the graph for efficient propagation of information, and will want analogues of Pearl's Back Door and Front Door theorems [40][41] for causal analysis.

It should be fairly clear from the above that most researchers looking to solve this problem have started with a BN, which as we know provides a highly successful way of describing and analysing symmetric processes. The Chain Event Graph, first proposed by Smith in [45], is developed instead from a Tree. Our graph can be used for problem representation (and is much less bulky than a Tree for large problems). We can interrogate it to read conditional independence properties (work on a full d-separation algorithm is ongoing). We have demonstrated the potential for information propagation and causal analysis in [61][60][62].

A probable reason that no-one has yet been completely successful in modifying the BN for use with asymmetric problems, is simply that analysis using a BN is variable-based. The CEG, as its name suggests, has an event-based topology, and analysis using a CEG is event-based, which makes it much more appropriate for asymmetric models. It owes its genesis to the set of asymmetric processes that occur in risk analysis, treatment regimes, biological regulation, emergency support systems [54], analysis of forensic evidence [43] etc. described above. In eliciting such models one often starts with an Event Tree (ET) [2][49], which is essentially a description of how the process unfolds rather than how the system might appear to an observer. Event Trees reflect model asymmetry both in model devel-

opment and in model sample space structure.

The CEG retains all of the ET’s convenience as a description of how things happen, whilst typically having far fewer edges and vertices. If a tree has sufficient symmetry that it can be represented by a BN, then the CEG derived from the tree also expresses all the conditional independence properties that are inherent in the BN; whilst if our model has insufficient symmetry to be adequately expressed as a BN, the CEG expresses (context-specific) conditional independence properties that would not be readily accessible in any other commonly used graphical representation.

The CEG is a mixed graph in that it contains both directed and undirected edges. Like the Tree (and unlike the Chain Graph [34][59]) the directed edges form root-to-leaf (in fact root-to-sink) paths corresponding to the atoms of the joint outcome space of the problem. Unlike the Tree, vertices may have more than one incoming edge, and vertices may be joined by undirected edges. It is these two properties that allow us to represent the conditional independencies of the model, as well as creating a graph less bulky than the original Tree.

1.4 Chain Event Graphs: Theory and application

In this thesis I concentrate on new developments relating to the construction and reading of CEGs, probability propagation on CEGs, and causal manipulation of CEGs. Chapter 2 describes the process by which we can construct a CEG. My approach throughout this thesis is to emphasise the key importance of the atomic events – the root-to-sink paths of the CEG, and as a result my description of CEG construction is significantly different from that in [45][53].

With a BN one can read *local* statements about near-adjacent vertices, and interrogate the graph about the possible independence of two sets of vertices given a third set, via the d-separation theorem (see for example [41][30]). In chapter 3 I develop methods for reading conditional independence properties off a CEG. Given the event-based topology of CEGs, the types of properties we can read go beyond the simple $X \perp\!\!\!\perp Y \mid Z$ that one can read off BNs, or $X \perp\!\!\!\perp Y \mid (Z = z)$ readable off context-specific BNs, to $X \perp\!\!\!\perp Y \mid \Lambda$ and $\Lambda_1 \perp\!\!\!\perp \Lambda_2 \mid \Lambda_3$, where the $\{\Lambda\}$ are events (ie. unions of atomic root-to-sink paths of the CEG).

Simple propagation on BNs [50][56][33] relies on the conditional independence properties of the BN to allow the joint distribution to be interpreted as a collection of *local* relationships between variables. BNs are usually triangularised before being converted to a

Junction Tree, and it is to this tree that the propagation algorithms are applied. The new CEG propagation algorithms described in chapter 4 are analogous to these Local Message Passing and Junction tree algorithms, and work in a manner which will be familiar to readers with knowledge of the BN-based algorithms. Short-cuts in the algorithms, analogous to the *Lazy* propagation of [35][6] are also described.

The algorithms are demonstrated using the blood condition example described in section 1.2. As this problem does not admit a natural Product Space, it cannot adequately be described by a BN, so conventional BN-based propagation algorithms are inappropriate here.

Causal manipulation as envisaged by Pearl [40][41] and others requires the existence of an *idle* or unmanipulated system upon which one then enacts an intervention. With a BN this intervention is commonly of the form $do\ X = x_0$ for some problem variable X and value x_0 ; and the manipulated BN is essentially the *idle* BN with edges entering the vertex X removed. Pearl's Back Door Theorem and Front Door Theorem [40][41] give conditions under which the general manipulated-probability expression for the model can be simplified by a reduction in the number of problem variables that need to be considered. This is very useful when some events cannot be observed or have a very high observational cost. What is particularly useful is that these conditions can be expressed graphically (ie. on the topology of the BN).

Chapter 5 starts with an explanation of what manipulating a CEG means, and how closely this relates to conditioning on a CEG. This close relationship is not restricted to analysis on CEGs, and this has not been commented upon in any depth before. It allows us much greater insight into what we mean by causal manipulation, and makes the techniques of causal analysis far more transparent. I then provide a Back Door Theorem for CEGs for general manipulations, before concentrating on *singular* manipulations, a class which contains the $do\ X = x_0$ interventions used with BNs as a subset. As with Pearl's Back Door Theorem I have been able to express one of the two conditions graphically (ie. on the topology of the CEG). I also provide a Front Door Theorem for CEGs for singular manipulations, which requires only two conditions as opposed to Pearl's three for his BN-based Front Door Theorem.

The threads running through chapters 2 to 5 are drawn together in chapter 6, where I give some idea of what Professor Smith and I will be working on in the near future.

2. Construction of Chain Event Graphs

2.1 Introduction and preliminary notation

I noted in section 1.2 that a Tree (an ET whose edges have been labelled with probabilities) can be used to completely describe an asymmetric model or the unfolding of an asymmetric process. Our CEG is a function of a Tree, so the construction of a CEG is a two-fold process: (1) construct a Tree which describes our model or process, (2) convert the Tree into a CEG. In this chapter I concentrate on the second stage, and I start by establishing some preliminary notation for working with both Trees and CEGs:

- Our basic building blocks (for both Trees and CEGs) are our atomic events which are single root-to-leaf paths (in our Tree or CEG). These will be denoted by λ , with some subscript (or superscript) when necessary.
- Compound events which are unions of our atomic events will be denoted by Λ , with some subscript (or superscript) when necessary.
- Where a number of events $\{\Lambda_i\}$ form a partition of the set of all atomic events, we may define some random variable with outcome space equal to $\{\Lambda_i\}$. Where appropriate, random variables may be given labels of the form A, B, C, \dots . The values taken by these variables will then be given labels of the form a, b, c, \dots , with some subscript when appropriate.

2.2 Trees, Events and Random Variables

In the following sections I describe the process by which we convert a Tree into a CEG. I then formally define the CEG. Thus consider the following motivating example: Suppose we conduct three experiments, the first of which has possible outcomes $\{a_0, a_1\}$, the second $\{b_0, b_1\}$, the third $\{c_0, c_1\}$. At this stage we are not concerned with any dependence between the outcomes of the three experiments. We can draw a Tree to represent the joint outcome space of these three experiments (edge labels here do not yet represent probabilities, but simply the unfolding of our experiments along each root-to-leaf path).

It would be conventional at this point to start by defining three random variables A, B, C which took values corresponding to the outcomes $\{a_0, a_1\}$, $\{b_0, b_1\}$, $\{c_0, c_1\}$. We could then draw a BN for this process if we knew the independence relationships between these three variables. The state space of this BN would be a Product Space $A \times B \times C$, and each atom of this space would be of the form (a_i, b_j, c_k) for $i, j, k = 0, 1$.

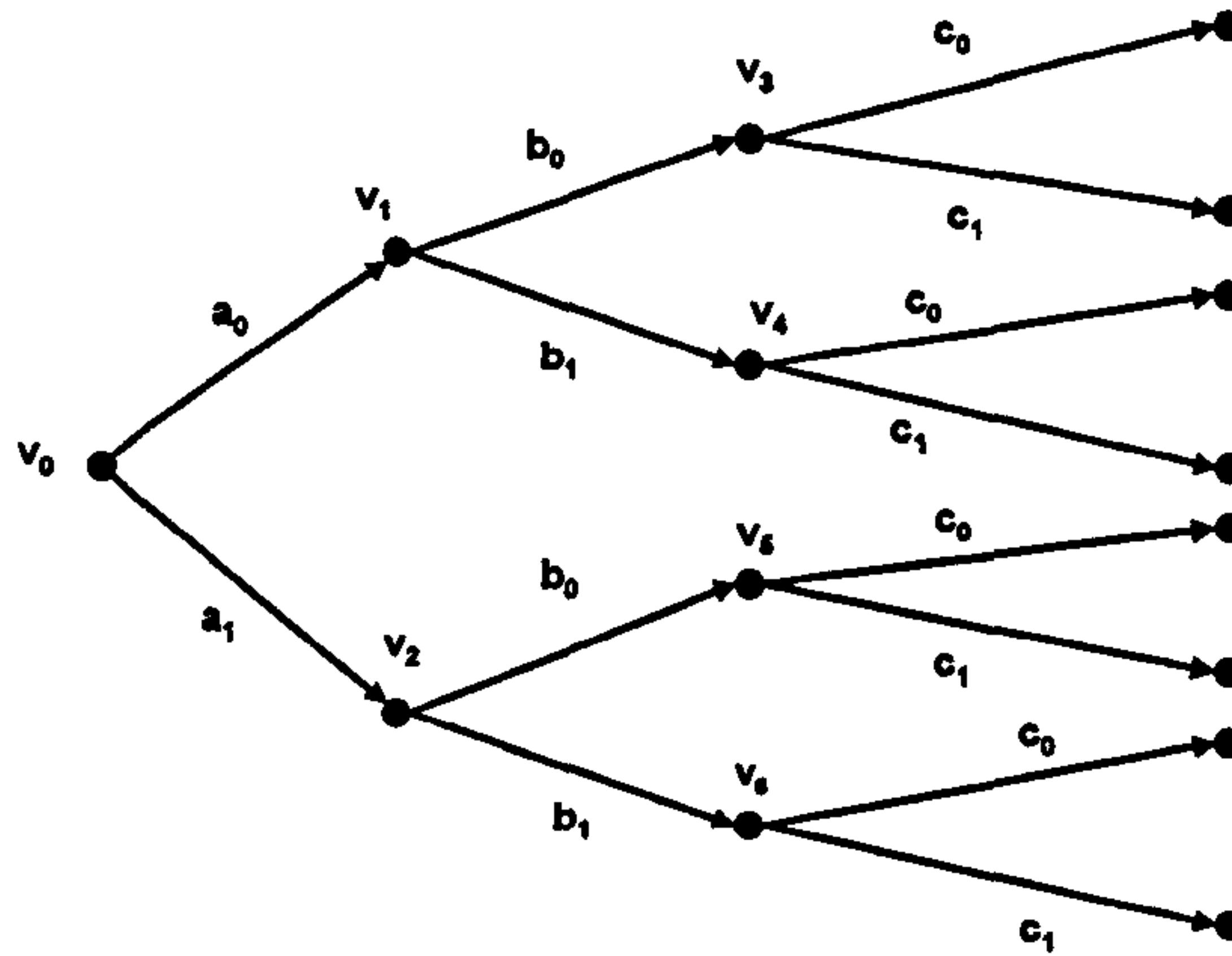


Figure 1: Tree for motivating example

Our process here is highly symmetric (at least until we put probabilities onto the tree), but most of the problems we might wish to represent via a CEG will not be, and as indicated in sections 1.1 and 1.2, the Product Space is not a natural choice of state space for asymmetric problems. As already noted in section 2.1, the fundamental building blocks for Trees and CEGs are not the measurement variables represented in BNs, but the atomic root-to-leaf paths $\{\lambda_i\}$, and the events created by forming unions of these items. We can still label the atoms as $a_i b_j c_k$, but this representation is simply a convenient label made up of the labels of the component edges of the path. In Figure 1, we have 8 atoms which can be labelled:

$$\begin{array}{ll}
 \lambda_1 : & a_0 b_0 c_0 \\
 \lambda_2 : & a_0 b_0 c_1 \\
 \lambda_3 : & a_0 b_1 c_0 \\
 \lambda_4 : & a_0 b_1 c_1 \\
 \lambda_5 : & a_1 b_0 c_0 \\
 \lambda_6 : & a_1 b_0 c_1 \\
 \lambda_7 : & a_1 b_1 c_0 \\
 \lambda_8 : & a_1 b_1 c_1
 \end{array}$$

From these atoms we can construct various compound events, some of which have obvious meanings in the context of our 3 experiments. So for example, we can define the events:

$$\begin{aligned}
 a_0 &= \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_4 \\
 b_0 &= \lambda_1 \cup \lambda_2 \cup \lambda_5 \cup \lambda_6 \\
 c_0 &= \lambda_1 \cup \lambda_3 \cup \lambda_5 \cup \lambda_7
 \end{aligned}$$

Definition 1: The event denoted a_i is defined to be the union of all atomic events λ_i which pass through an edge which has the label a_i .

It is important to be clear as to what is meant when we use a label such as a_0 , as it can be used as an outcome of an experiment, a label for an edge on our tree, part of the label of an atomic event λ_i , and as the label for an event as in Definition 1. Care obviously needs to be taken, but a quick look at the context that the symbol a_0 is used in will always tell us which meaning we are using.

Only now having defined our events do we introduce random variables, and we construct these from our compound events. So we define the random variable A to take values corresponding to the outcomes $\{a_0, a_1\} = \{\lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_4, \lambda_5 \cup \lambda_6 \cup \lambda_7 \cup \lambda_8\}$, and B to take values corresponding to the outcomes $\{b_0, b_1\} = \{\lambda_1 \cup \lambda_2 \cup \lambda_5 \cup \lambda_6, \lambda_3 \cup \lambda_4 \cup \lambda_7 \cup \lambda_8\}$. Note that we are now abusing notation slightly by allowing a_0 to also denote the outcome taken by a random variable A . Again, if we are careful this should not cause any problems. Having defined our events and random variables we can obviously introduce probabilities to our model:

We assign probabilities (summing to 1) to each of our atomic events, so:

$$\pi_1 = \pi(\lambda_1) = \pi(a_0 b_0 c_0) \quad \text{etc.}$$

and from these calculate probabilities for our compound events, using the fact that each compound event is a union of disjoint atoms, so, for example:

$$\pi(a_0) = \pi_1 + \pi_2 + \pi_3 + \pi_4$$

$$\pi(b_0) = \pi_1 + \pi_2 + \pi_5 + \pi_6$$

We now have all the necessary material for constructing our CEG, and we can commence the process. As already noted, any disjoint set of events whose summed probability is 1 can be used to define a random variable, and it is very useful at this point to create random variables to help us with the CEG-construction process as follows:

Consider a random variable X_1 taking values corresponding to the outcomes $\{b_0, b_1\}$, but which takes these values **not** with probabilities $\pi(b_0)$ and $\pi(b_1)$, but with probabilities:

$$\pi_{X_1}(b_0) = \pi(b_0 \mid a_0)$$

$$\pi_{X_1}(b_1) = \pi(b_1 \mid a_0)$$

We have called this variable X_1 as it is associated with the vertex v_1 , but note that in this example X_1 has the distribution of the conditional random variable $B|a_0$.

It is clearly the case that having defined the probabilities of compound events, we can label edges with probabilities. So for example the vertex v_1 carries the atomic events

$\lambda_1, \lambda_2, \lambda_3, \lambda_4$, and the edge $e(v_1, v_3)$ carries the atomic events λ_1, λ_2 , so this edge can be labelled with the probability:

$$\begin{aligned}\pi(\lambda_1 \cup \lambda_2 \mid \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_4) &= \pi(\lambda_1 \cup \lambda_2 \cup \lambda_5 \cup \lambda_6 \mid \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_4) \\ &= \pi(b_0 \mid a_0)\end{aligned}$$

Similarly the vertex v_3 carries the atomic events λ_1, λ_2 , and the upper edge leaving v_3 carries the atomic event λ_1 , so this edge can be labelled with the probability:

$$\begin{aligned}\pi(\lambda_1 \mid \lambda_1 \cup \lambda_2) &= \pi(\lambda_1 \cup \lambda_3 \cup \lambda_5 \cup \lambda_7 \mid \lambda_1 \cup \lambda_2) \\ &= \pi(c_0 \mid a_0 b_0)\end{aligned}$$

where $a_0 b_0$ is the event which is the union of all atomic events which pass through both an edge labelled a_0 and an edge labelled b_0 .

The first use to which we put the variables $\{X_i\}$ is to assess conditional independence relationships existing between the variables A, B and C .

We can clearly define X_3 to take values corresponding to the outcomes $\{c_0, c_1\}$ and

$$\begin{aligned}\pi_{X_3}(c_0) &= \pi(c_0 \mid a_0 b_0) \\ \pi_{X_3}(c_1) &= \pi(c_1 \mid a_0 b_0)\end{aligned}$$

X_3 is a variable associated with the vertex v_3 , and in this example has the distribution of the conditional variable $C \mid (a_0 b_0)$.

If we have for example $\pi_{X_3}(c_0) = \pi_{X_5}(c_0)$ ($\Leftrightarrow \pi_{X_3}(c_1) = \pi_{X_5}(c_1)$), then this tells us that:

$$C \amalg A \mid b_0 \tag{2.2.1}$$

$$\begin{aligned}\pi_{X_3}(c_0) &= \pi_{X_5}(c_0) \\ \Rightarrow \pi(c_0 \mid a_0 b_0) &= \pi(c_0 \mid a_1 b_0) \\ \Rightarrow \frac{\pi_1}{\pi_1 + \pi_2} &= \frac{\pi_5}{\pi_5 + \pi_6} \\ \Rightarrow \frac{\pi_1}{\pi_1 + \pi_2} &= \frac{\pi_5}{(\pi_1 + \pi_2 + \pi_5 + \pi_6) - (\pi_1 + \pi_2)} \\ \Rightarrow \pi_1(\pi_1 + \pi_2 + \pi_5 + \pi_6) - \pi_1(\pi_1 + \pi_2) &= \pi_5(\pi_1 + \pi_2) \\ \Rightarrow \frac{\pi_1}{\pi_1 + \pi_2} &= \frac{\pi_1 + \pi_5}{\pi_1 + \pi_2 + \pi_5 + \pi_6} \\ \Rightarrow \pi(c_0 \mid a_0 b_0) &= \pi(c_0 \mid b_0)\end{aligned}$$

If also $\pi_{X_4}(c_0) = \pi_{X_6}(c_0)$ we then have $C \amalg A \mid B$.

As most of our problems will be asymmetric, most of our independence properties will be context-specific like expression (2.2.1), and these are investigated in the next chapter. Having labelled our edges with probabilities, we can now look at groups of vertices and use the variables $\{X_i\}$ as described to simplify labels (so here for example we would replace $c_0 \mid a_0 b_0$ by $c_0 \mid b_0$).

We can also define variables Y_1 and Y_2 as follows:

Y_1 takes values corresponding to the outcomes $\{b_0 c_0, b_0 c_1, b_1 c_0, b_1 c_1\}$ (where $b_0 c_0$ etc. are events defined analogously with $a_0 b_0$ above) with:

$$\pi_{Y_1}(b_0 c_0) = \pi(b_0 c_0 \mid a_0)$$

$$\pi_{Y_1}(b_0 c_1) = \pi(b_0 c_1 \mid a_0)$$

Y_2 has outcome space as that of Y_1 , with:

$$\pi_{Y_2}(b_0 c_0) = \pi(b_0 c_0 \mid a_1)$$

Clearly Y_1 is a variable associated with the vertex v_1 , and in this example has the distribution of the conditional variable $(B \times C) \mid a_0$. As noted earlier however, when we consider asymmetric structures it is unlikely that our outcome space will be a Product Space, and hence describing random variables as $(B \times C) \mid a_0$ or even $B \times C$ will be meaningless.

In a sense we can use our $\{X_i\}$ to analyse *local* conditional independencies of the form $C \perp\!\!\!\perp A \mid b_0$. Our $\{Y_i\}$ can be used to analyse more *global* properties, so for example, if our Tree in Figure 1 was extended by performing a further three binary experiments (with corresponding random variables D, E, F), and if we noted that $\pi_{Y_3}(c_i, d_j, e_k, f_l) = \pi_{Y_3}(c_i, d_j, e_k, f_l)$ for $i, j, k, l = 0, 1$, then we could deduce that $(C, D, E, F) \perp\!\!\!\perp A \mid b_0$.

Having introduced our new variables X_i and Y_i , we are now in a position to consider the asymmetric processes for which our CEG will be used. Consider the Tree in Figure 2. Here the second experiment doesn't happen if the result of the first experiment is a_1 .

Our atoms λ_i are defined by the root-to-leaf paths, and it is straightforward to show that we can define events called a_0, a_1, c_0 and c_1 with:

$$\pi(a_0) + \pi(a_1) = 1$$

$$\pi(c_0) + \pi(c_1) = 1$$

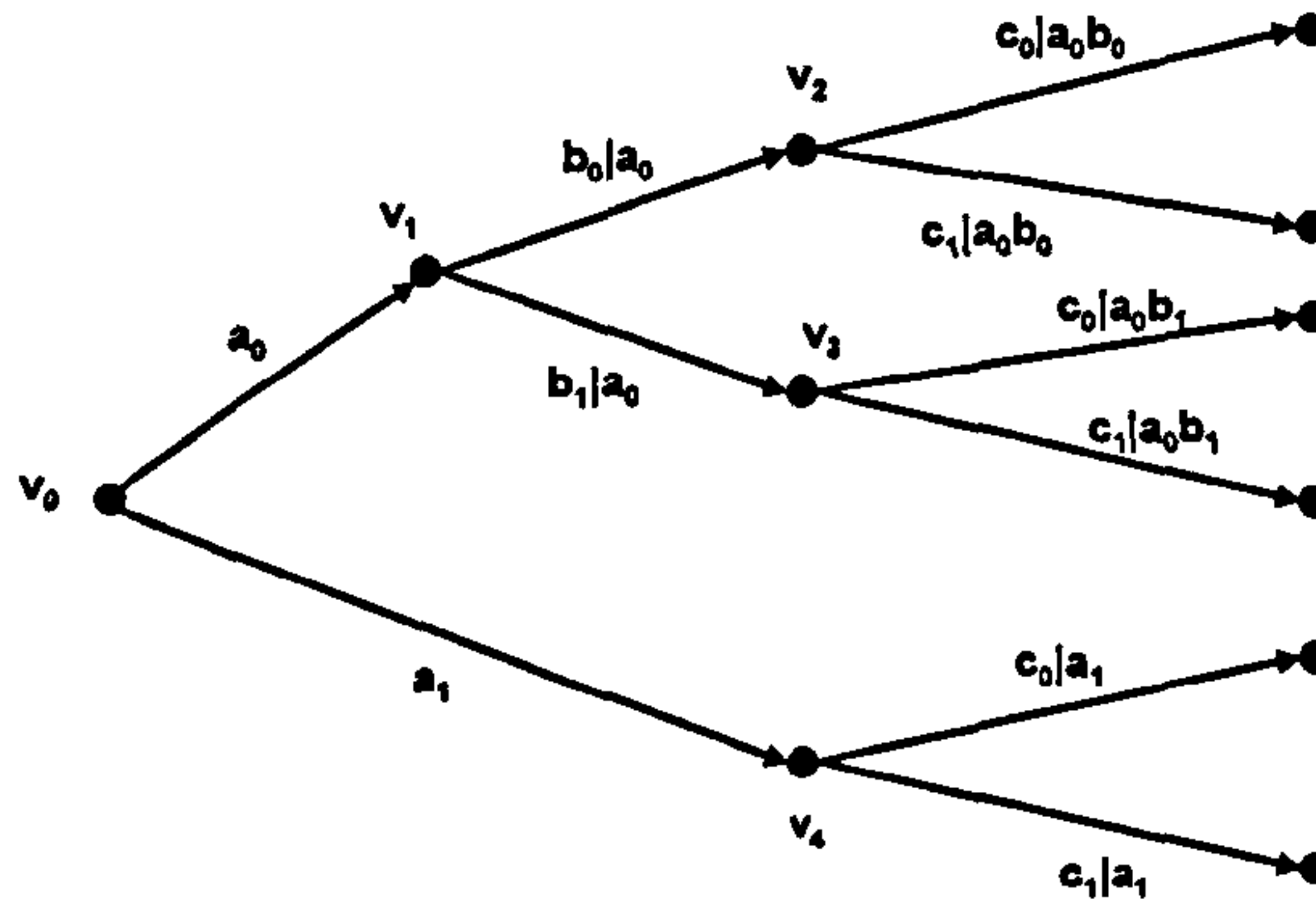


Figure 2: A Probability Tree with some asymmetry

NB: Edge labels do not reflect any as yet undisclosed dependence between the variables

But if we define the events b_0 and b_1 as in Definition 1, we see that $\pi(b_0) + \pi(b_1) \neq 1$. In fact:

$$\pi(b_0) + \pi(b_1) + \pi(a_1) = 1$$

It is none-the-less reasonable to say that there is a random variable B whose sample space is the outcome space of the second experiment, if we imagine this experiment to have taken place. We can clearly still define X_1 precisely as before, and note that it has the distribution of the conditional random variable $B|a_0$. As noted in section 1.3, many examples in risk analysis [55], decision analysis [15], biological regulation [7] etc. exhibit this sort of asymmetry.

2.3 Extending the model

In the previous section I added probabilities to an Event Tree to produce a Probability Tree. A Probability Tree T is defined by its vertex set $V(T)$, its edge set $E(T)$ and its edge-probabilities. In this thesis I follow the convention that the Tree is a function both of the Event Tree and the probability distribution imposed on it, and hence that each edge-probability is greater than zero. This will mean, that when I come to define a CEG, all edge-probabilities on this will also be greater than zero. It would be a fruitful line of research to consider what might happen if this condition were to be relaxed. In this scenario, edges with zero probabilities would represent outcomes that existed under some probability distributions but not under the one currently imposed. In chapter 4, when we update a CEG following observation of an event we *prune* edges with zero probabilities. These would be retained if we were to relax this condition.

Definition 2: For a Probability Tree model $T(V(T), E(T))$, let $V^I \subset V(T)$ be the set of non-root, non-leaf vertices of the Tree (I for intermediate).

In this thesis I follow Shafer [49] in that our Probability Tree is essentially an embellishment of an Event Tree (we add edge-probabilities), and hence simply a graphical description of how a process unfolds. It is convenient to think of this process as a sequence of experiments, so as in the example in section 2.2, let our model consist of a sequence of n experiments, with n associated *criterion variables* $\{A, B, \dots N\}$. In this thesis I impose the condition that all root-to-leaf paths (in Trees) or root-to-sink paths (in CEGs) utilise edges corresponding to the criterion variables in the **same order** $(A, B, \dots N)$, whilst not precluding the possibility that some paths pass through **no** edges corresponding to some or other variable. Moreover, I also impose the condition that all root-to-leaf or root-to-sink paths utilise a maximum of one edge corresponding to any criterion variable. These conditions can be relaxed, and there is some discussion throughout this chapter of the possible consequences of so doing.

To each vertex $v_i \in \{v_0\} \cup V^I$ assign a criterion variable from the set above. The same criterion variable will usually be assigned to more than one v_i . Label the edges leaving v_i with outcomes from the outcome space of the criterion variable associated with v_i . Note that there may be outcomes in this space with no corresponding edge leaving v_i . Using the analogy of a sequence of experiments, this would correspond to outcomes to an experiment that could not happen given a particular history of outcomes to the previous experiments.

If there are root-to-leaf paths in our Tree which have fewer than n edges then there will be criterion variables (like B in the asymmetric Tree in section 2.2) whose outcome spaces do not partition the set $\{\lambda_i\}$.

Definition 3: For a Probability Tree model $T(V(T), E(T))$ and:

- (i) $v \in V(T)$, define $\Lambda(v)$ as the event which is the union of all root-to-leaf paths that pass through the vertex v .
- (ii) $e \in E(T)$, define $\Lambda(e)$ as the event which is the union of all root-to-leaf paths that pass through the edge e .

Definition 4: For a Probability Tree model $T(V(T), E(T))$, if the vertex $v_i \in \{v_0\} \cup V^I$ has associated criterion variable D (say), taking values corresponding to the outcomes

$\{d_0, d_1, \dots, d_{n(D)}\}$, define the random variable X_i taking values corresponding to the outcomes $\{d_0, d_1, \dots, d_{n(D)}\}$ with probabilities:

$$\pi_{X_i}(d_j) = \pi(d_j \mid \Lambda(v_i))$$

Note that if there does not exist an edge labelled d_k (say) leaving the vertex v_i , then $\pi_{X_i}(d_k) = 0$. Note also that X_i (as D) may not have an outcome space which partitions $\{\lambda_i\}$.

It should be noted that any criterion variable E (say) whose outcome space does not partition $\{\lambda_i\}$ can be made to do so by the addition of an event e_ϕ to its outcome space, where:

$$\pi(e_\phi) = 1 - \sum_{j=0}^{n(E)} \pi(e_j)$$

and which can be thought of as the event that the E th experiment is not performed.

If the vertex v_j has associated criterion variable E note that $\pi_{X_j}(e_\phi) = 0$.

Definition 5: For vertex v_i with criterion variable D (say), the subset of the atomic events consisting of all root-to-leaf paths utilising an edge labelled d_j ($j = 0, 1, \dots, n(D)$) is partitioned by the random variable Y_i into events labelled $d_j e_k \dots n_z$ ($j = 0, 1, \dots, n(D)$, $k = 0, 1, \dots, n(E), \phi, \dots, z = 0, 1, \dots, n(N), \phi$), where $d_j e_k \dots n_z$ labels the union of all paths λ which pass through edges labelled d_j, e_k, \dots and n_z .

For convenience, we let (for example) $d_1 e_1 \dots f_1 g_\phi h_1 \dots n_1$ denote the union of all paths λ which pass through edges labelled $d_1, e_1, \dots, f_1, h_1, \dots, n_1$ and pass through no edges labelled g_j for any j .

Y_i takes values corresponding to the events formed by this partition with probabilities:

$$\pi_{Y_i}(d_j e_k \dots n_z) = \pi(d_j e_k \dots n_z \mid \Lambda(v_i))$$

Note that each $d_j e_k \dots n_z$ is simply a label for an event, and that the outcome space of Y_i is not specified as a Product space.

Example 1:

So in our Tree in Figure 2, we have:

$$\Lambda(v_0) = \Omega$$

$$\Lambda(v_1) = (a_0)$$

$$\Lambda(v_2) = (a_0 b_0)$$

$$\Lambda(v_3) = (a_0 b_1)$$

$$\Lambda(v_4) = (a_1)$$

X_0 takes values corresponding to $\{a_0, a_1\}$ with probabilities $\pi_{X_0}(a_i) = \pi(a_i)$.

X_1 takes values corresponding to $\{b_0, b_1\}$ with probabilities $\pi_{X_1}(b_i) = \pi(b_i \mid a_0)$.

X_2 takes values corresponding to $\{c_0, c_1\}$ with probabilities $\pi_{X_2}(c_i) = \pi(c_i \mid a_0 b_0)$.

Y_0 takes values corresponding to $\{a_0 b_0 c_0, a_0 b_0 c_1, a_0 b_1 c_0, a_0 b_1 c_1, a_0 b_\phi c_0, a_0 b_\phi c_1, a_1 b_0 c_0,$

$a_1 b_0 c_1, a_1 b_1 c_0, a_1 b_1 c_1, a_1 b_\phi c_0, a_1 b_\phi c_1\}$

with probabilities $\pi(a_0 b_0 c_0), \pi(a_0 b_0 c_1), \pi(a_0 b_1 c_0), \pi(a_0 b_1 c_1), 0$ (since $\pi(b_\phi \mid a_0) = 0$),

$0, 0$ (since $\pi(b_0 \mid a_1) = 0$), $0, 0, 0, \pi(a_1 c_0), \pi(a_1 c_1)$.

If we only list outcomes with non-zero probabilities, we get, for example:

Y_1 takes values corresponding to $\{b_0 c_0, b_0 c_1, b_1 c_0, b_1 c_1\}$

with probabilities $\pi(b_0 c_0 \mid a_0), \pi(b_0 c_1 \mid a_0), \pi(b_1 c_0 \mid a_0), \pi(b_1 c_1 \mid a_0)$.

2.4 Trees and CEGs

The process of deriving the CEG from our Tree has been described in [45][53][61][60][62], but I intend to approach it in a radically different way. Part of the motivation for my approach in this chapter comes from the response we had to the descriptions given in [61][60]. What we have here is clearly a *process*, a fact that is nearly always obscured by a formal definition. We found that listeners to the talk at the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems where this process was described had a clearer understanding than those who had simply read the paper in [61]. Consequently I intend to motivate the formal definition in section 2.5 by first describing the process in detail. This description is lengthy as the mathematics underlying the definition of a CEG is subtle. However the process described is actually very straightforward and quick, as can be seen in Example 4 in section 2.6.

The descriptions in [45][61][60][62] make use of maps between subtrees and between the Tree and the CEG. As already indicated I have chosen to approach the subject differently, and I intend to use our variables $\{X_i\}$ and $\{Y_i\}$ to effect the conversion of the Tree into a CEG. I start with an informal introduction indicating the three basic steps in the conversion:

- Suppose the edges leaving the vertices $\{v_j\}$ in the Tree can be labelled by those leaving a vertex v_1 (in that the edges correspond to the same value of a criterion variable, and carry the same probability) then these vertices are all joined by undirected edges.
- If the subtrees rooted in any two vertices of the Tree are identical (both in topology and in probability distribution), then these two vertices are combined into a single vertex.

In practice it is often sensible to do these two steps the other way round, as it saves joining vertices by edges when they are later going to be combined into a single vertex.

- All leaf-vertices are combined into one sink-vertex, and edges carrying the same outcomes with identical probabilities are given the same colour.

Note that we do not need to know the probability distribution over the Tree — we only need to be aware of when subtrees have the same distribution or when edges have the same probability; we do not need to know what these probabilities are.

We now consider these steps in more detail:

Step 1: We first augment our Tree with a set of undirected edges. I call the resultant graph an Augmented Tree.

Definition 6: A pair of vertices $v_i, v_j \in V^I$ ($i \neq j$) are *stage-equivalent* iff the variables X_i and X_j have the same outcome space and the same probability distribution over this space.

We write $X_i \equiv X_j$.

Definition 7: The Augmented Tree $T^A(V(T^A), E_d(T^A), E_u(T^A))$ derived from a Tree $T(V(T), E(T))$ is a mixed graph with $V(T^A) = V(T)$, $E_d(T^A) = E(T)$, and $e(v_i, v_j) \in E_u(T^A)$ if $v_i, v_j \in V(T)$ are stage-equivalent under Definition 6.

$E_u(T^A)$ is a set of undirected edges (so stage-equivalent vertices are connected by undirected edges in the Augmented Tree).

Step 2:

Definition 8: A pair of vertices $v_i, v_j \in V^I$ ($i \neq j$) are *equivalent* iff the variables Y_i and Y_j have the same outcome space and the same probability distribution over this space.

We write $Y_i \equiv Y_j$.

NB: If v_i and v_j are equivalent then they are necessarily stage-equivalent, and hence connected by an undirected edge in our Augmented Tree.

Using the conditions imposed in section 2.3, we can start to make substantive statements about the topology of our Tree.

Proposition 1:

Let $T(V(T), E(T))$ and $\{A, B, \dots, N\}$ be a Probability Tree model and a set of criterion variables, where (I) the existence of an edge in $E(T)$ implies an edge-probability greater

than zero; (II) all root-to-leaf paths in T utilise edges corresponding to the criterion variables in the same order, and no root-to-leaf path utilises more than one edge corresponding to any criterion variable. Then, if Y_i and Y_j have the same outcome space and the same probability distribution over this space, then the sub-trees rooted in the vertices v_i and v_j are identical both in their topology and in the probabilities labelling their component edges.

Note that this is **not** true if we allow different paths to pass through the criterion variables in different orders, as can be seen in Example 2.

Example 2:

Consider the Probability Tree in Figure 3.

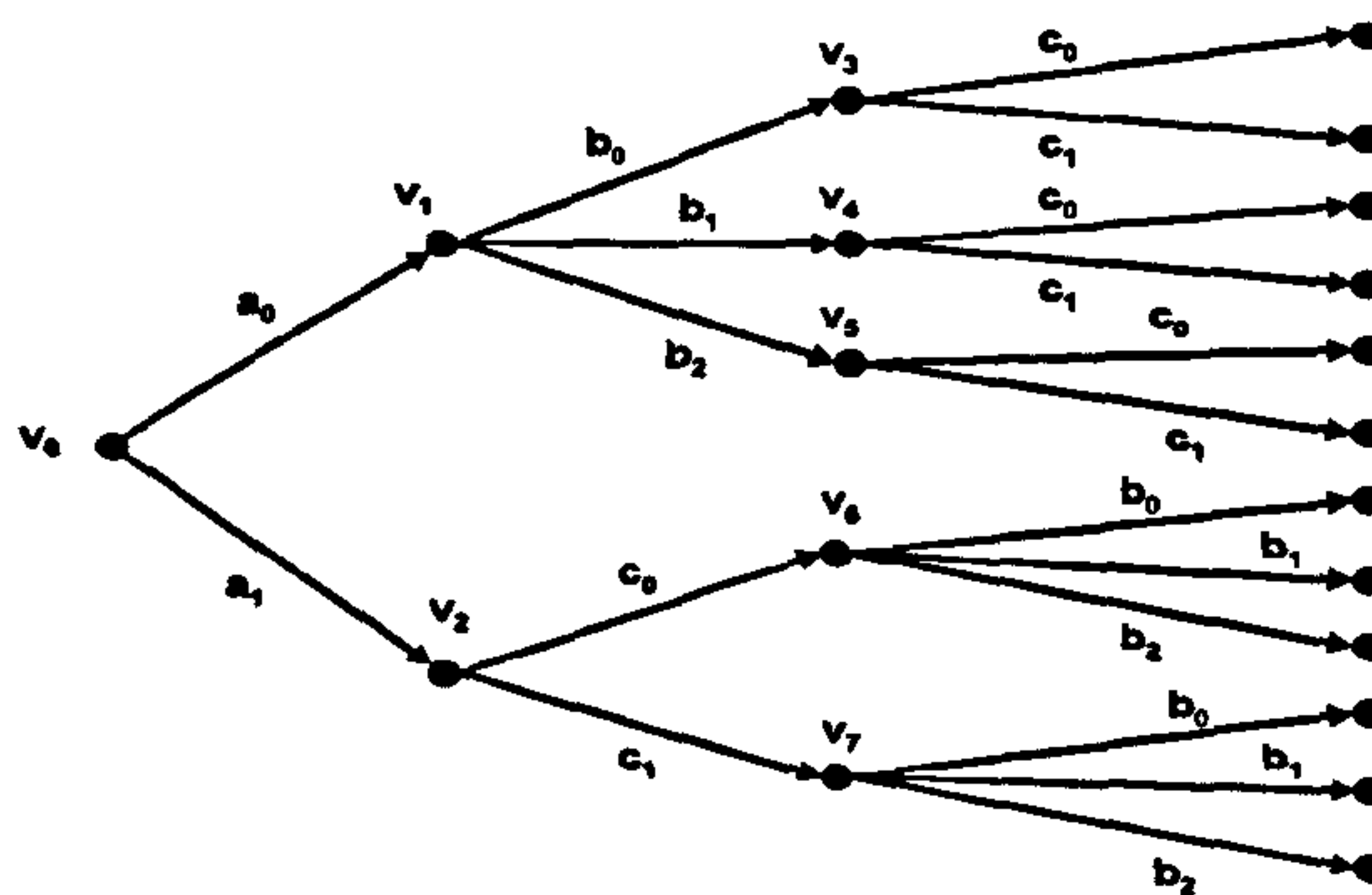


Figure 3: Probability Tree for Example 2

The variable Y_1 takes values corresponding to the events $\{b_0c_0, b_0c_1, b_1c_0, b_1c_1, b_2c_0, b_2c_1\}$, the variable Y_2 takes values corresponding to the events $\{c_0b_0, c_0b_1, c_0b_2, c_1b_0, c_1b_1, c_1b_2\}$, but by our definitions, the event b_0c_0 is the same event as c_0b_0 , so Y_1 and Y_2 have the same outcome space.

If we define the variables A, B, C in the obvious manner, and let $A \amalg (B, C)$, $B \amalg C$, then the edge-labels in Figure 3 can be read as probabilities, and clearly Y_1 and Y_2 have the same probability distribution over their common outcome space. But the subtrees rooted in the vertices v_1 and v_2 are not identical in topology.

Clearly in this case, since $B \amalg C$ it would be a simple matter to reorder the paths in Figure 3 so that the condition (II) held, and hence the result of Proposition 1 held. So in this example, the restrictions imposed by condition (II) are not overly severe.

Proof of Proposition 1:

- (1) Consider vertices v_i and v_j with criterion variable E . Using (I) and (II) we have that there exists a $v_0 \rightarrow v_i \rightarrow \text{leaf}$ path utilising edges labelled e_k, f_l, \dots, n_z in that order

if and only if $\pi(e_k f_l \dots n_z \mid \Lambda(v_i)) \neq 0$. If Y_i and Y_j have the same outcome space and the same probability distribution over this space, then $\pi(e_k f_l \dots n_z \mid \Lambda(v_i)) \neq 0$ if and only if $\pi(e_k f_l \dots n_z \mid \Lambda(v_j)) \neq 0$, if and only if there exists a $v_0 \rightarrow v_j \rightarrow \text{leaf}$ path utilising edges labelled e_k, f_l, \dots, n_z in that order.

As the edges leaving any vertex in the Tree are labelled with outcomes from the outcome space of the criterion variable associated with that vertex, no two outgoing edges from a vertex can have the same label. Hence the subtrees rooted in v_i and v_j have the same topology.

(2) Suppose $\pi(e_k f_l \dots n_z \mid \Lambda(v_i)) \neq 0$ for a specified k , and some values l, m, \dots, z .

Then $Y_i \equiv Y_j$ implies $\pi(e_k f_l \dots n_z \mid \Lambda(v_i)) = \pi(e_k f_l \dots n_z \mid \Lambda(v_j))$, which with the fact that the subtrees rooted in v_i and v_j have the same topology, gives us that $\sum_{l,m,\dots,z} \pi(e_k f_l \dots n_z \mid \Lambda(v_i)) = \sum_{l,m,\dots,z} \pi(e_k f_l \dots n_z \mid \Lambda(v_j))$.

That is, the probability of leaving v_i via an edge labelled e_k equals the probability of leaving v_j via an edge labelled e_k .

Consider also that $\sum_{m,n,\dots,z} \pi(e_k f_l \dots n_z \mid \Lambda(v_i)) = \sum_{m,n,\dots,z} \pi(e_k f_l \dots n_z \mid \Lambda(v_j))$, and hence the probability of leaving v_i via edges labelled e_k and f_l equals the probability of leaving v_j via edges labelled e_k and f_l , which implies that the probabilities associated with the edges f_l in the subtrees rooted in v_i and v_j are identical.

We can now use induction to demonstrate that the subtrees rooted in v_i and v_j have identical probabilities labelling their component edges.

It occasionally happens that a sequence of experiments occurs in different orders on different paths. There is a good example of this in [55]. If we stick rigourously to the idea that our Tree is an exact description of how the process unfolds, then this might appear to cause problems with some of our definitions and with Proposition 1. In fact, in this context it is often sensible to define different criterion variables associated with the experiments happening at different points on different root-to-leaf paths, which sidesteps the problem. Also, if we move away from the idea that the Tree is an exact description of how the process unfolds, it is usually possible (as seen in Example 2) to reorder trees (and their resultant CEGs), without affecting the conditional independence structure of the model they represent. This has an analogy in BN-analysis, where several different BNs can display the same conditional independence structure. Smith in [45][53] uses alternative definitions of the CEG to allow consideration of CEGs with different orderings on different paths.

Note that Definitions 6 and 8, together with the conditions upon which Proposition 1 is

predicated, have some interesting consequences:

- (i) No two vertices on the same root-to-leaf path can be equivalent or stage-equivalent.
- (ii) There cannot exist vertices v_a, v_b, v_i, v_j with v_i preceding v_j on some root-to-leaf path, and v_b preceding v_a on some root-to-leaf path, such that v_i, v_a are equivalent (stage-equivalent) and v_j, v_b are equivalent (stage-equivalent).

Note that if we allow different orderings on different root-to-sink paths in a CEG, and hence different orderings on different root-to-leaf paths in a Tree then (ii) is no longer valid, at least for stage-equivalence. The validity of (i) depends on our disallowing any root-to-leaf path from utilising more than one edge corresponding to one criterion variable, which in turn depends on us disallowing the same experiment being performed more than once. Again, there are arguments for not imposing this condition, but equally one can argue that if an experiment is performed a second time it is necessarily a **different** experiment, so should be associated with a different criterion variable.

If we accept (i) and (ii), then there is a significant further consequence: There exist natural **partial** orderings of the sets of equivalent vertices in any Augmented Tree.

Step 2 continued: We say that a vertex v which is not equivalent to any other vertex forms a set of equivalent vertices by itself, and we also define the set $W_0 = \{v_0\}$.

Following one of the partial orderings described above, label the sets of equivalent vertices in our Augmented Tree W_1, W_2, W_3, \dots . For each set W_i choose (arbitrarily if W_i contains more than one vertex) one vertex $v \in W_i$ and relabel this vertex w_i .

What follows over the next page is a rigorous description of the process by which the Augmented Tree is converted into a Chain Event Graph. The formal definition of a CEG is much shorter as can be seen in section 2.5; the actual practical process of conversion is really straightforward (as can be seen in the examples at the end of this chapter) and can be summarised by the 2nd (and 3rd) bullet point at the start of this section.

Consider all edges from v_0 ($\in W_0$) to a $v_j \in W_1$. The edge from v_0 to w_1 remains unaltered, but all other of these edges (if any exist) are replaced by $v_0 - w_1$ edges.

It will now be the case that all $v_j \in W_1$ except w_1 have no incoming edges.

We now remove from our Augmented Tree all vertices $v_j \in W_1$ except w_1 . We also remove all vertices and edges on paths leaving any $v_j \in W_1$ (except those leaving w_1) ie. we remove the sub-trees rooted in any $v_j \in W_1$ ($v_j \neq w_1$). We further remove any

undirected edges which have at least one end connected to a vertex which has now been removed.

Now consider all paths from $v_0 (= w_0)$ to a vertex $v_j \in W_2$. These will be of at most two types:

- (i) $v_0 - v_j \in W_2$ edges
- (ii) $v_0 - w_1 - v_j \in W_2$ paths

- (i) If there exists a $v_0 - w_2$ edge, this remains unaltered. All $v_0 - v_j$ ($v_j \in W_2, v_j \neq w_2$) edges (if any exist) are replaced by a $v_0 - w_2$ edge.
- (ii) If there exists a $w_1 - w_2$ edge, this remains unaltered. All $w_1 - v_j$ ($v_j \in W_2, v_j \neq w_2$) edges (if any exist) are replaced by a $w_1 - w_2$ edge.

It will now be the case that all $v_j \in W_2$ except w_2 have no incoming edges. Remove from our Augmented Tree all vertices $v_j \in W_2$ (except w_2) and sub-trees rooted in these v_j as before.

The general stage in this process is:

Consider all paths from $v_0 (= w_0)$ to a vertex $v_j \in W_k$. These paths may well be of different lengths, but will all consist of edges joining vertices labelled $w_0, w_{a(\lambda)}, \dots, w_{n(\lambda)}$ (with $a(\lambda), \dots, n(\lambda) \in \{1, 2, \dots, k-1\}$), and an edge joining $w_{n(\lambda)}$ to either w_k or some $v_j \in W_k$ ($v_j \neq w_k$).

If this edge joins $w_{n(\lambda)}$ to w_k , then it remains unaltered. If it joins $w_{n(\lambda)}$ to $v_j \in W_k$ ($v_j \neq w_k$), it is replaced by an edge from $w_{n(\lambda)}$ to w_k . All $v_j \in W_k$ except w_k now have no incoming edges, so remove all vertices $v_j \in W_k$ except w_k , and the sub-trees rooted in these v_j . Also remove any undirected edges which have at least one end connected to a vertex which has now been removed.

Continue this process for all sets W_1, W_2, \dots in our partial ordering.

Any w_k, w_l which before relabelling were connected by an undirected edge in our Augmented Tree remain connected by an undirected edge in our modified Tree.

Alternatively, if $v_i \in W_k$ was connected to $v_j \in W_l$ by an undirected edge in our Augmented Tree, then w_k is connected to w_l by an undirected edge in our modified Tree.

Step 3:

All leaf-vertices are combined into a single sink-vertex, labelled w_∞ .

Note that the root-vertex has already been relabelled w_0 , and that all intermediate vertices are now labelled w_i in such a way that if w_i precedes w_j on any path within our graph then $i < j$.

If two vertices v_i, v_j in our Augmented Tree are stage-equivalent then our Tree-based variables X_i and X_j have the same outcome space, and the edges emanating from v_i, v_j can be identified with the elements of this space. Moreover, any element x (say) of this outcome space is such that $\pi_{X_i}(x) = \pi_{X_j}(x)$.

If a vertex w_k in our modified Tree corresponds to a vertex v_i in our Augmented Tree, then it has the same number of edges emanating from it, and these edges can be identified with the elements of the outcome space of X_i .

So, if two vertices w_k, w_l in our modified Tree are connected by an undirected edge, then they correspond to two vertices v_i, v_j (say) in our Augmented Tree, such that $X_i \equiv X_j$. Hence the edges emanating from w_k and w_l can both be identified with the elements of the outcome space of X_i (X_j), and moreover these elements are such that $\pi_{X_i}(x) = \pi_{X_j}(x)$ for any x in the outcome space. We label the *corresponding* edges (those identified with the same element of the outcome space of X_i (X_j)) emanating from w_k and w_l with the same colour.

The resulting graph is called a Chain Event Graph. The vertices $\{w_i\}$ are called *positions*.

Collections of positions which are connected to each other by undirected edges are called *stages*. Individual positions which are unconnected to any other positions by undirected edges are deemed also to be *stages*. We show below that positions connected by undirected edges are stage-equivalent in broadly the same sense as stage-equivalent vertices in our Tree.

It should be noted that every path in the original Tree corresponds to a path in the CEG, and there are no paths in the CEG that do not correspond to a path in the original Tree. Hence the atomic events of our model are precisely the $w_0 \rightarrow w_\infty$ paths in our CEG.

We are now in a position to create random variables for use with CEGs, and as long as we make it clear whether we are discussing Trees or CEGs we can use the same letters to describe variables which have broadly similar functions in the two forms.

We know that for any $v_i, v_j \in W_k$, the Tree-based variables Y_i, Y_j have the same outcome space and the same probability distribution over this space, so our choice of which $v_j \in W_k$ to relabel as w_k is completely arbitrary (providing there is more than one vertex $v_j \in W_k$). So, in our CEG, if the position w_k corresponds to a vertex $v_j \in W_k$, we can create a CEG-based variable Y_k having the same outcome space as the Tree-based Y_j and the same probability distribution over this space.

Also, a pair of positions w_s, w_t in our CEG are connected by an undirected edge whenever a pair of corresponding vertices v_i, v_j are connected by an undirected edge in our Augmented Tree, which occurs when the Tree-based variables X_i and X_j have the same outcome space and the same probability distribution over this space. This means that if for each position w_k (with corresponding vertex v_j) we create a CEG-based variable X_k which has the same outcome space as the Tree-based X_j and the same probability distribution over this space, then if two positions w_s, w_t are connected by an undirected edge, then we have $X_s \equiv X_t$, and we call this property *stage-equivalence* as before.

2.5 A formal definition of a Chain Event Graph

Definition 9: An uncoloured Chain Event Graph derived from a Probability Tree $T(V(T), E(T))$ is the mixed graph $C(W(C), E_d(C), E_u(C))$, where $W(C), E_d(C)$ and $E_u(C)$ are defined as follows:

- (1) The set of non-leaf vertices of T ($\{v_0\} \cup V^I$) is partitioned into equivalence classes by the Tree-based variables $\{Y_i\}$. Write $Y_i \equiv Y_j$ if Y_i and Y_j have the same outcome space and the same probability distribution over this space.

Let $W_0 = \{v_0\}$. For vertices $v_i, v_j \in V^I$, if $Y_i \equiv Y_j$ let $v_i, v_j \in W_k$ for some k , and if $Y_i \not\equiv Y_j$ let v_i, v_j be members of different equivalence classes.

Let there be N such equivalence classes excluding W_0 .

For each W_k , arbitrarily choose a vertex $w_k \in W_k$. Then $W(C) = \left[\bigcup_{k=0}^N w_k \right] \cup w_\infty$.

- (2A) If for any $v_i \in \{v_0\} \cup V^I, v_i \in W_m$, there exist q edges from v_i to vertices $\{v_j\} \in W_n$, then there exist q directed edges $e(w_m, w_n) \in E_d(C)$ ($q \geq 0$).
- (2B) If for any $v_i \in \{v_0\} \cup V^I, v_i \in W_m$, there exist q edges from v_i to leaf-vertices in $V(T)$, then there exist q directed edges $e(w_m, w_\infty) \in E_d(C)$ ($q \geq 0$).
- (3) V^I is partitioned into (stage) equivalence classes by the Tree-based variables $\{X_i\}$. Write $X_i \equiv X_j$ if X_i and X_j have the same outcome space and the same probability distribution over this space.

If there exist vertices $v_i, v_j \in V^I$, with $v_i \in W_m$, $v_j \in W_n$ ($m \neq n$) and $X_i \equiv X_j$, then there exists an undirected edge $e(w_m, w_n) \in E_u(C)$.

Henceforth we will, if necessary relabel the positions $\{w_k\}$ so that if w_m precedes w_n on any $w_0 \rightarrow w_\infty$ path, then $m < n$.

Definition 10: If a position $w_k \in W(C)$ has been relabelled from a vertex $v_i \in V(T)$, then we define the CEG-based random variable Y_k to have the same outcome space as the Tree-based random variable Y_i , and the same probability distribution over this space.

Definition 11: If a position $w_k \in W(C)$ has been relabelled from a vertex $v_i \in V(T)$, then we define the CEG-based random variable X_k to have the same outcome space as the Tree-based random variable X_i , and the same probability distribution over this space.

Definition 12: $W(C) \setminus \{w_\infty\}$ is partitioned into (stage) equivalence classes by the CEG-based random variables $\{X_i\}$. These equivalence classes are called *stages* and denoted $\{u_j\}$.

Positions $w_m, w_n \in W(C) \setminus \{w_\infty\}$ which belong to the same stage are called *stage-equivalent*, and are such that X_m, X_n have the same outcome space and the same probability distribution over this space. If $w_m, w_n \in u_j$ for some stage u_j , we write $X_m \equiv X_n$. Note that w_m and w_n will be connected by an undirected edge $e(w_m, w_n) \in E_u(C)$.

We can define a coloured CEG as follows:

Definition 13: A coloured Chain Event Graph derived from a Probability Tree $T(V(T), E(T))$ is the mixed graph $C(W(C), E_d(C), E_u(C))$ with coloured edges, where $W(C), E_d(C)$ and $E_u(C)$ are defined as in Definition 9, and:

(4) For any $w \in W(C)$ we define $\Lambda(w)$ as the event which is the union of all $w_0 \rightarrow w_\infty$ paths that pass through the position w .

For any $e \in E_d(C)$ we define $\Lambda(e)$ as the event which is the union of all $w_0 \rightarrow w_\infty$ paths that pass through the edge e .

Let $\{e_m^i\}$ be the set of edges in $E_d(C)$ emanating from the position w_m , and let e_m^x be the edge in this subset corresponding to the element x in the outcome space of X_m . That is:

$$\pi(\Lambda(e_m^x) \mid \Lambda(w_m)) \triangleq \pi_{X_m}(x)$$

Then if there exists an edge $e(w_m, w_n) \in E_u(C)$,

$$\pi(\Lambda(e_m^x) \mid \Lambda(w_m)) = \pi(\Lambda(e_n^x) \mid \Lambda(w_n))$$

and the edges e_m^x and e_n^x have the same colour in C .

It is worth noting that whether we follow the process described in section 2.4 or use the definition in this section, we can always reconstruct our original Tree from our CEG.

The ideas of equivalence and stage-equivalence are quite subtle, and can be interpreted in slightly different ways. When we write, for example $X_i \equiv X_j$, do we mean that these variables are the same or just that they have the same distribution? In the context of Definitions 10 and 11, I suggest that as Y_i (X_i) and Y_k (X_k) are defined on different structures then we must mean the latter, but what about the cases in Definitions 6, 8 and 12?

From Definition 4, if we have Tree-based variables X_i and X_j such that $X_i \equiv X_j$, then X_i and X_j may have the same outcome space and the same distribution over this space, but they are defined differently:

$$\pi_{X_i}(x) = \pi(x \mid \Lambda(v_i))$$

$$\pi_{X_j}(x) = \pi(x \mid \Lambda(v_j)) \quad \text{where } \Lambda(v_j) \neq \Lambda(v_i)$$

and we write $X_i \equiv X_j$ only if $\pi(x \mid \Lambda(v_i)) = \pi(x \mid \Lambda(v_j))$ for all elements x of the sample space. So I prefer the latter interpretation. The same argument holds for the Tree-based variables $\{Y_i\}$ and the CEG-based variables $\{X_i\}$. Milan Studeny (November 2006) prefers the alternative interpretation, but in this thesis, when I write, for example, $X_i \equiv X_j$, I mean that these variables have the same distribution. One of the reasons for my choice comes from considering what might happen if we analysed the CEG when conditioned on or manipulated to an event (chapters 4 and 5). It is inappropriate to discuss this in detail here, but note that both these operations can destroy stage-equivalence in a CEG, so that $X_i \equiv X_j$ in the idle CEG does not imply $X_i \equiv X_j$ in the conditioned or manipulated CEG.

2.6 The construction process in practice

We include here examples from [60] and [61]. These examples are used to illustrate the process by which we construct a CEG, and to show how fast the process is in practice.

Example 3:

This example (mentioned in section 1.2) is concerned with the treatment and effects of a particular genetic blood condition.

A population is screened for this particular genetic blood condition, and divides into four groups:

- n : Patient does not have the blood condition (–ve)
- x : Patient does have the blood condition (+ve) and has blood group O
- y : +ve and blood group B
- z : +ve and blood group A or AB

Patients in groups y or z are also checked to see whether they are Rhesus –ve (m) or +ve (m'). Note that the probability of being Rhesus +ve is the same whether a patient is in group y or group z .

In each of groups x , y and (z, m') the condition affects the patient in one of three ways:

- r : The condition has no effect on the patient at all
- s : Affects daily life, but patient stable, and no effect on life expectancy
- t : Affects daily life, requires treatment

Membership of these groups is governed by:

$$\pi(r | x) = \pi(r | y, m') = \pi(r | z, m') \neq \pi(r | y, m)$$

$$\pi(s | x) = \pi(s | y, m') = \pi(s | z, m') \neq \pi(s | y, m)$$

$$\pi(t | x) = \pi(t | y, m') = \pi(t | z, m') \neq \pi(t | y, m)$$

Patients in groups r and s have the same life expectancy as those in n . Patients in groups (y, t) , (z, m', t) and (z, m) have the same life expectancy, but this is different from that of a patient in group (x, t) , which is in turn different from those in r , s or n .

This scenario can be represented by the Tree in Figure 4, where d is some form of binary life expectancy indicator (such as whether or not the patient survives twelve months). As can be seen from the Tree, the process can be thought of as a series of four experiments with associated random variables:

- A with outcome space $\{n, x, y, z\}$
- B with outcome space $\{m, m'\}$
- C with outcome space $\{r, s, t\}$
- D with outcome space $\{d, d'\}$

Notice that the 2nd and 3rd experiments do not always happen.

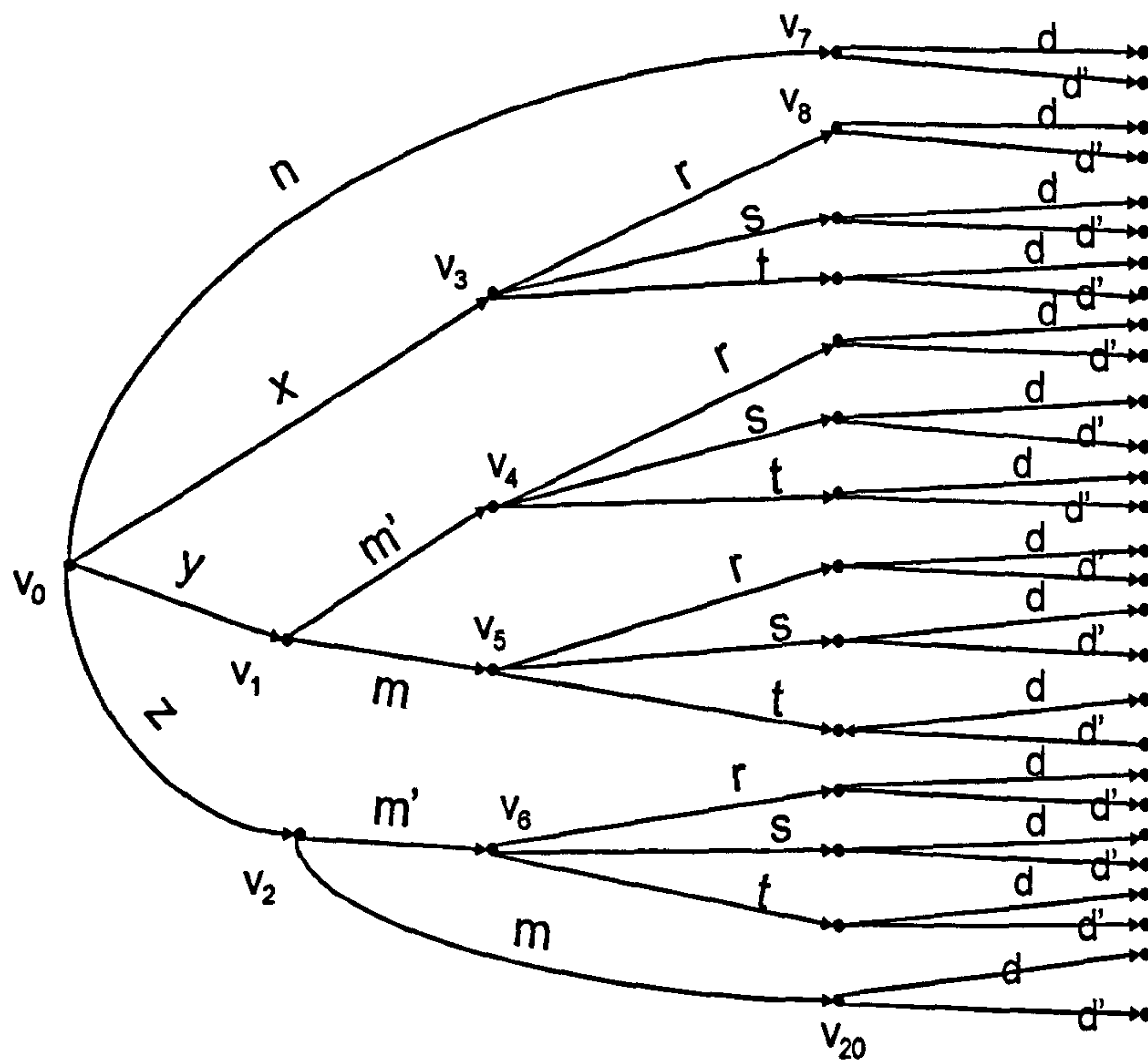


Figure 4: Tree for Blood Example

Before converting this Tree into a CEG we need to add the probability information that we know, and this is shown in Figure 5. If we now follow the construction process outlined in section 2.4, we get:

Step 1:

Define the variables X_0, \dots, X_{20} . So for example:

X_1 and X_2 take values corresponding to $\{m, m'\}$ with probabilities

$$\begin{aligned} \pi_{X_1}(m) &= \pi(m \mid y) = \pi(m \mid y \text{ or } z) \\ \pi_{X_1}(m') &= \pi(m' \mid y) = \pi(m' \mid y \text{ or } z) \\ \pi_{X_2}(m) &= \pi(m \mid z) = \pi(m \mid y \text{ or } z) \\ \pi_{X_2}(m') &= \pi(m' \mid z) = \pi(m' \mid y \text{ or } z) \end{aligned}$$

X_3 takes values corresponding to $\{r, s, t\}$ with probabilities

$$\begin{aligned} \pi_{X_3}(r) &= \pi(r \mid x) = \pi(r \mid x \text{ or } ym' \text{ or } zm') \\ \pi_{X_3}(s) &= \pi(s \mid x) = \pi(s \mid x \text{ or } ym' \text{ or } zm') \\ \pi_{X_3}(t) &= \pi(t \mid x) = \pi(t \mid x \text{ or } ym' \text{ or } zm') \end{aligned}$$

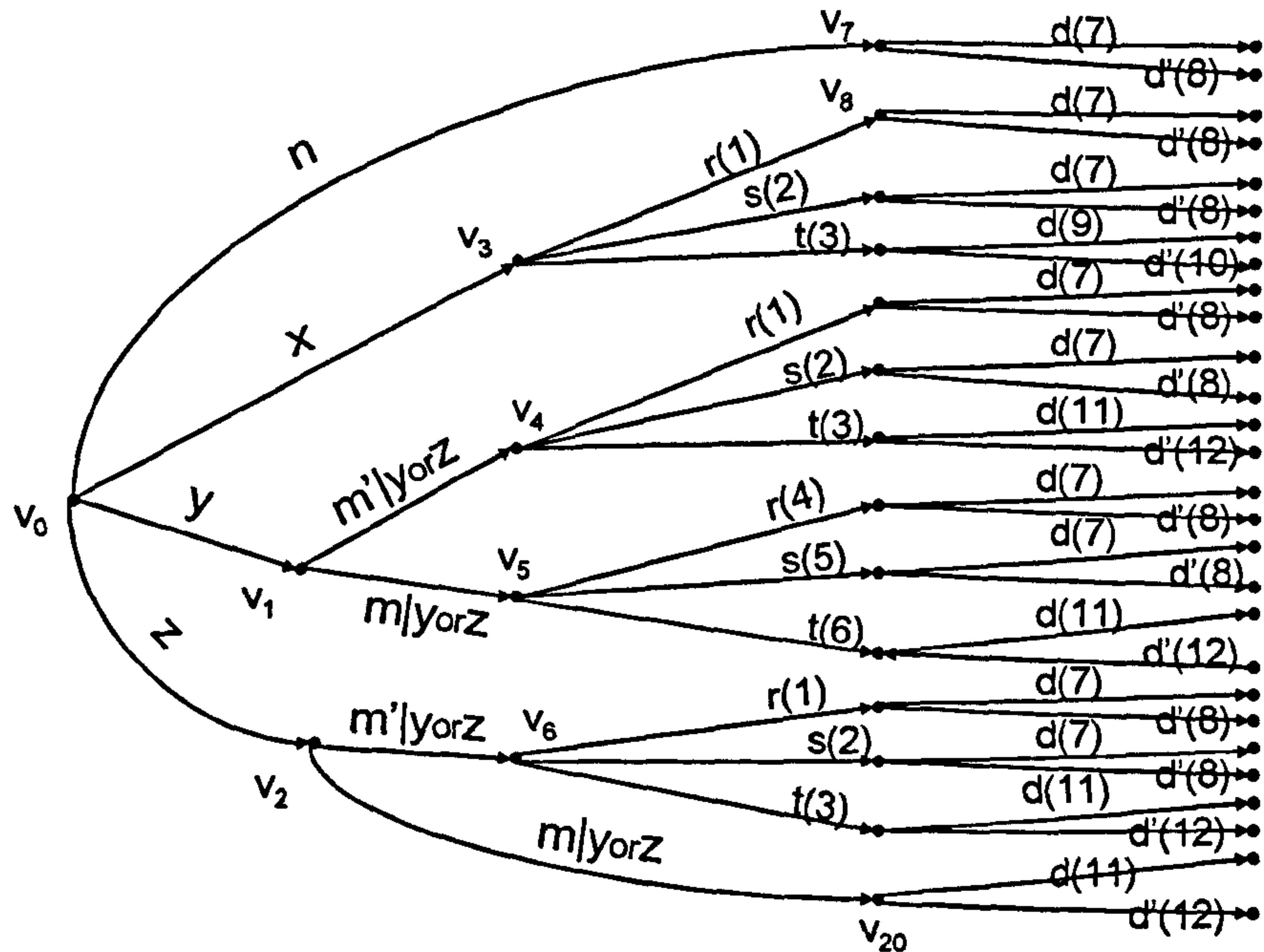
We then construct our Augmented Tree by connecting any pair of vertices (v_i, v_j) such that $X_i \equiv X_j$, with an undirected edge. So we connect by undirected edges:

$$(v_1, v_2) \quad \text{Note that } X_1 \equiv X_2 \text{ from above}$$

(v_3, v_4, v_6)

$(v_7, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18})$

$(v_{13}, v_{16}, v_{19}, v_{20})$



1: $r \mid x \text{ or } ym' \text{ or } zm'$	7: $d \mid n \text{ or } r \text{ or } s$
2: $s \mid x \text{ or } ym' \text{ or } zm'$	8: $d' \mid n \text{ or } r \text{ or } s$
3: $t \mid x \text{ or } ym' \text{ or } zm'$	9: $d \mid xt$
4: $r \mid ym$	10: $d' \mid xt$
5: $s \mid ym$	11: $d \mid yt \text{ or } zm't \text{ or } zm$
6: $t \mid ym$	12: $d' \mid yt \text{ or } zm't \text{ or } zm$

Figure 5: Tree for Blood Example showing patient groups and equivalent probabilities

It is important to note that we do not actually need to define any of our variables X_i or check whether any pair $X_i \equiv X_j$! All we really need to do is follow the steps as described in the bullet points at the start of section 2.4, and all our actions can be deduced from the Tree in Figure 5. For example, we can read from the Tree that the edges emanating from v_1 and v_2 carry identical probabilities so we can connect them with an undirected edge (etc). Our Augmented Tree is shown in Figure 6.

Step 2:

Following the construction process outlined in section 2.4, we now define the variables Y_0, \dots, Y_{20} . So for example, Y_1 takes values corresponding to $\{mrd, mrd', msd, msd', mtd,$

$$mtd', mc_\phi d, mc_\phi d', m'rd, m'rd', m'sd, m'sd', m'td, m'td', m'c_\phi d, m'c_\phi d'\}$$

with probabilities $\frac{1}{2}$ and $\frac{1}{2}$ to only one edge from a leaf vertex and either

$$\pi_{Y_1}(mrd) = \pi(mrd \mid y)$$

$$\pi_{Y_1}(mrd') = \pi(mrd' \mid y)$$

$$\pi_{Y_1}(msd) = \pi(msd \mid y)$$

$$\pi_{Y_1}(msd') = \pi(msd' \mid y)$$

$$\pi_{Y_1}(mtd) = \pi(mtd \mid y)$$

$$\pi_{Y_1}(mtd') = \pi(mtd' \mid y)$$

$$\pi_{Y_1}(m'rd) = \pi(m'rd \mid y) = \pi(m'rd \mid y \text{ or } z)$$

$$\pi_{Y_1}(m'rd') = \pi(m'rd' \mid y) = \pi(m'rd' \mid y \text{ or } z)$$

$$\pi_{Y_1}(m'sd) = \pi(m'sd \mid y) = \pi(m'sd \mid y \text{ or } z)$$

$$\pi_{Y_1}(m'sd') = \pi(m'sd' \mid y) = \pi(m'sd' \mid y \text{ or } z)$$

$$\pi_{Y_1}(m'td) = \pi(m'td \mid y) = \pi(m'td \mid y \text{ or } z)$$

$$\pi_{Y_1}(m'td') = \pi(m'td' \mid y) = \pi(m'td' \mid y \text{ or } z)$$

$$\pi_{Y_1}(mc_\phi d) = \pi_{Y_1}(mc_\phi d') = \pi_{Y_1}(m'c_\phi d) = \pi_{Y_1}(m'c_\phi d') = 0$$

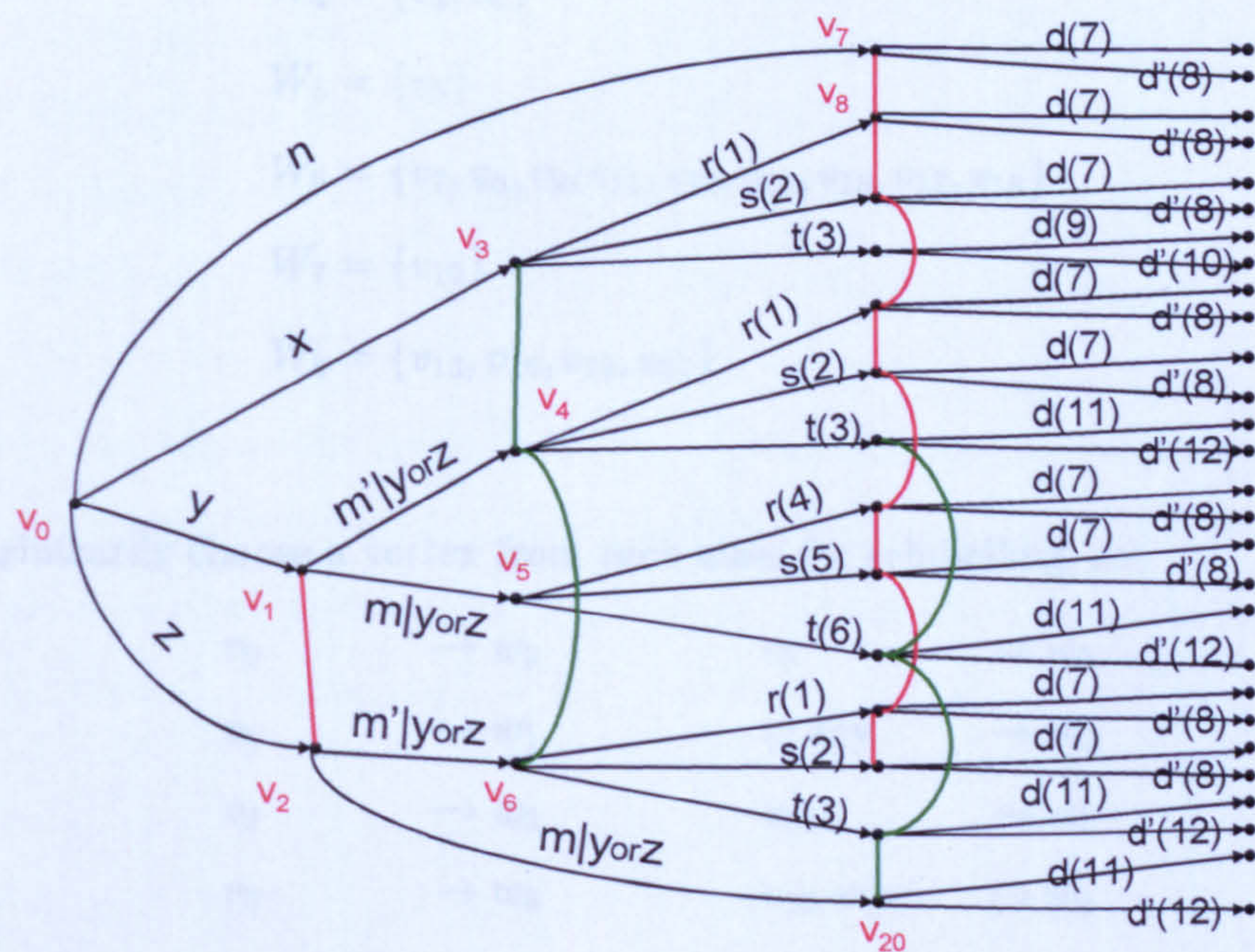


Figure 6: Augmented Tree for Blood Example

We now look for pairs Y_i such that $Y_i \equiv Y_j$, and if we do this we find:

$$Y_4 \equiv Y_6$$

$$Y_7 \equiv Y_8 \equiv Y_9 \equiv Y_{11} \equiv Y_{12} \equiv Y_{14} \equiv Y_{15} \equiv Y_{17} \equiv Y_{18}$$

$Y_{13} \equiv Y_{16} \equiv Y_{19} \equiv Y_{20}$ (see also paragraph, we get the graph in Figure 7.

Note that the 2nd and 3rd sets of equivalences here could be deduced immediately from the facts that v_7, v_8, \dots, v_{20} lie only one edge from a leaf-vertex and that:

$$\begin{aligned} X_7 &\equiv X_8 \equiv X_9 \equiv X_{11} \equiv X_{12} \equiv X_{14} \equiv X_{15} \equiv X_{17} \equiv X_{18} \\ X_{13} &\equiv X_{16} \equiv X_{19} \equiv X_{20} \end{aligned}$$

Following the construction process we identify 9 equivalence classes, and a partial order, so for example:

$$\begin{aligned} W_0 &= \{v_0\} \\ W_1 &= \{v_1\} \\ W_2 &= \{v_2\} \\ W_3 &= \{v_3\} \\ W_4 &= \{v_4, v_6\} \\ W_5 &= \{v_5\} \\ W_6 &= \{v_7, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18}\} \\ W_7 &= \{v_{10}\} \\ W_8 &= \{v_{13}, v_{16}, v_{19}, v_{20}\} \end{aligned}$$

We now arbitrarily choose a vertex from each class for relabelling, so:

v_0	$\rightarrow w_0$	v_5	$\rightarrow w_5$
v_1	$\rightarrow w_1$	v_7 say	$\rightarrow w_6$
v_2	$\rightarrow w_2$	v_{10}	$\rightarrow w_7$
v_3	$\rightarrow w_3$	v_{20} say	$\rightarrow w_8$
v_4 say	$\rightarrow w_4$		

The edges $v_0 (w_0) \rightarrow v_1 (w_1)$, $v_0 (w_0) \rightarrow v_2 (w_2)$ and $v_0 (w_0) \rightarrow v_3 (w_3)$ remain unaltered. The edge $v_1 (w_1) \rightarrow v_4 (w_4)$ remains unaltered, but the edge $v_2 (w_2) \rightarrow v_6$ is replaced by an edge $w_2 (v_2) \rightarrow w_4$. The vertex v_6 and the sub-tree rooted in v_6 are removed (we call this process *pruning* as it is in effect removing branches from a Tree).

If we continue the process of redirecting edges and pruning, we get the graph in Figure 7.

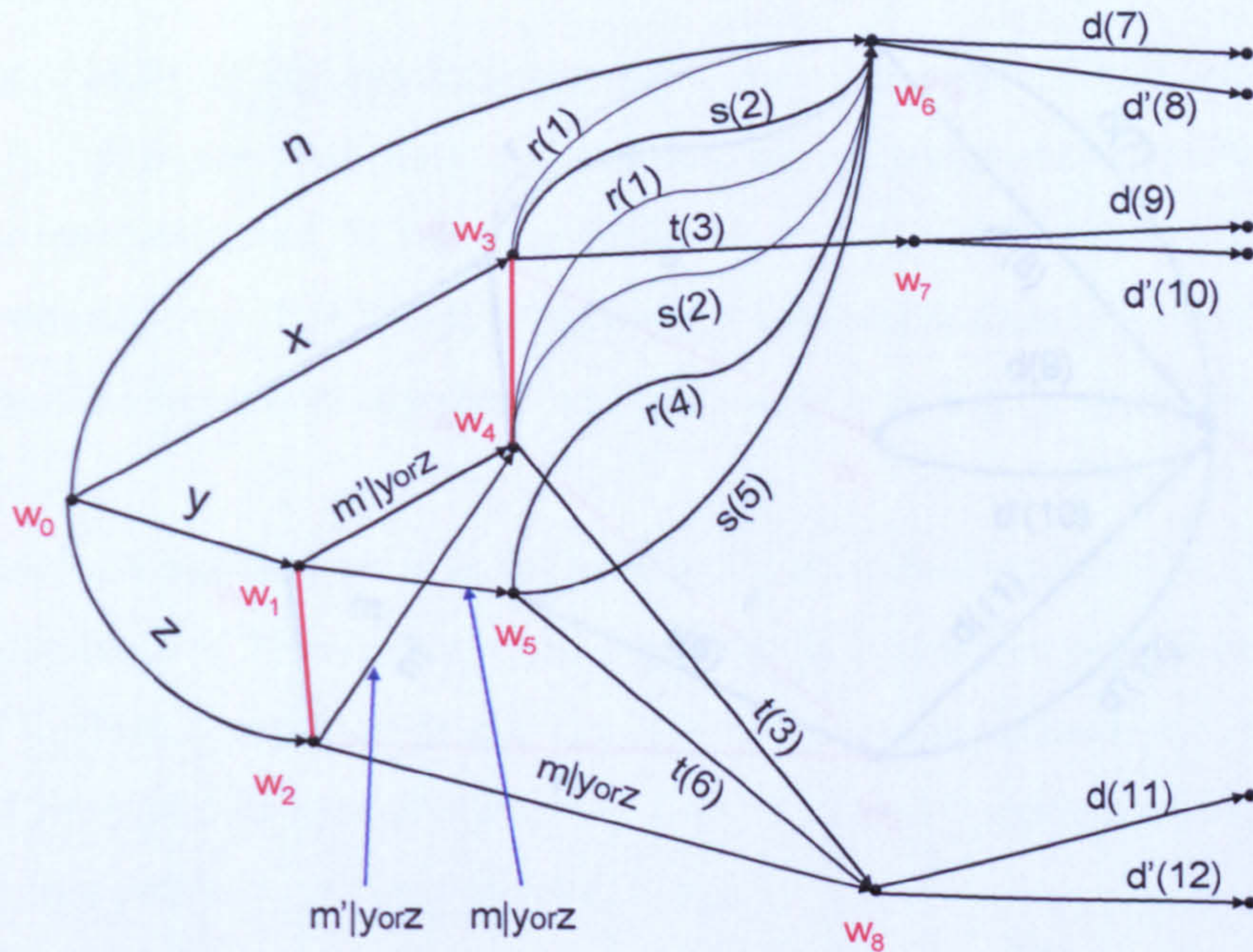


Figure 7: Blood Example graph following Step 2 of the construction process

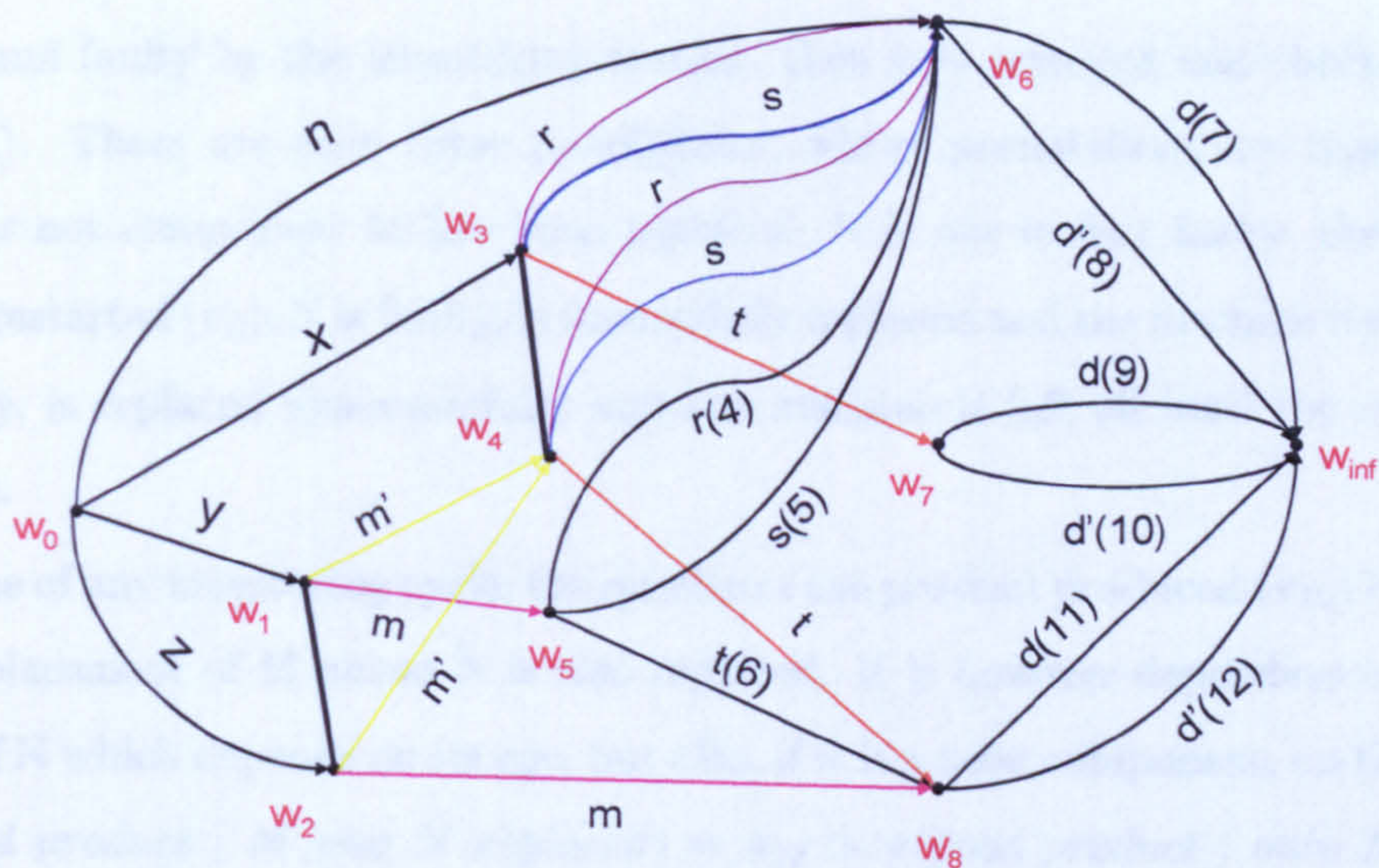
It is important to note again that we do not actually need to define any of our variables Y_i nor check whether any pair $Y_i \equiv Y_j$, nor follow the detailed instructions given in section 2.4. All we need to do is follow the instructions given in the second bullet point at the start of section 2.4, and all our actions can be deduced from the Augmented Tree in Figure 6.

For example, looking at the Augmented Tree we notice that the sub-trees rooted in v_4 and v_6 are identical in both topology and the probabilities on their edges, and hence these two vertices can be combined into a single vertex. Similarly for $\{v_7, v_8, v_9, v_{11}, v_{12}, v_{14}, v_{15}, v_{17}, v_{18}\}$ and for $\{v_{13}, v_{16}, v_{19}, v_{20}\}$.

Step 3:

All leaf-vertices are combined into a single sink-vertex w_∞ , and colours are added. So for example, w_3 and w_4 are connected by an undirected edge; the edges emanating from w_3 and w_4 represent the outcomes r, s and t ; and the probabilities $\pi(r | x)$, $\pi(s | x)$ and $\pi(t | x)$ are identical to the probabilities $\pi(r | ym' \text{ or } zm')$, $\pi(s | ym' \text{ or } zm')$ and $\pi(t | ym' \text{ or } zm')$; so the corresponding edges leaving w_3 and w_4 are given the same colour.

The final CEG is given in Figure 8.



- 1: $r \mid x \text{ or } ym' \text{ or } zm'$
 - 2: $s \mid x \text{ or } ym' \text{ or } zm'$
 - 3: $t \mid x \text{ or } ym' \text{ or } zm'$
 - $m' \mid y \text{ or } z$
 - $m \mid y \text{ or } z$
- 4,5,6,7,8,9,10,11 & 12 as for Figure 4

Figure 8: Final CEG for Blood Example

Example 4:

This example is concerned with a fault-monitoring process in a production line.

A machine in a production line utilises two replaceable components M and N. Faults in these components do not automatically cause the machine to fail, but do affect the quality of the product, so the machine incorporates an automated monitoring system, which is completely reliable for finding faults in M, but which can detect a fault in N when it is functioning correctly.

In any monitoring cycle, component M is checked first, and there are three initial possibilities: M, N checked and no faults found (π_1 on the Tree in Figure 9); M checked, fault found, machine switched off (π_2); M checked, no fault found, N checked, fault found, machine switched off (π_3).

If M is found faulty it is replaced and the machine switched back on (vertex v_1), and N is then checked. N is then either found not faulty (π_4), or faulty and the machine

switched off (π_5).

If N is found faulty by the monitoring system, then it is removed and checked (vertices v_2 and v_3). There are then three possibilities, whose probabilities are independent of whether or not component M has been replaced: N is not in fact faulty, the machine is reset and restarted (π_6); N is faulty, is successfully replaced and the machine restarted (π_7); N is faulty, is replaced unsuccessfully and the machine is left off until the engineer can see it (π_8).

At the time of any monitoring cycle, the quality of the product produced (π_{10}) is unaffected by the replacement of M unless N is also replaced. It is however dependent on the *effectiveness* of N which depends on its *age*, but also, if it is a new component, on the *age* of M; so: $\pi(\text{good product} \mid M \text{ and } N \text{ replaced}) = \pi_{12} > \pi(\text{good product} \mid \text{only } N \text{ replaced}) = \pi_{14} > \pi(\text{good product} \mid N \text{ not replaced}) = \pi_{10}$.

This scenario can be represented by the Tree in Figure 9.

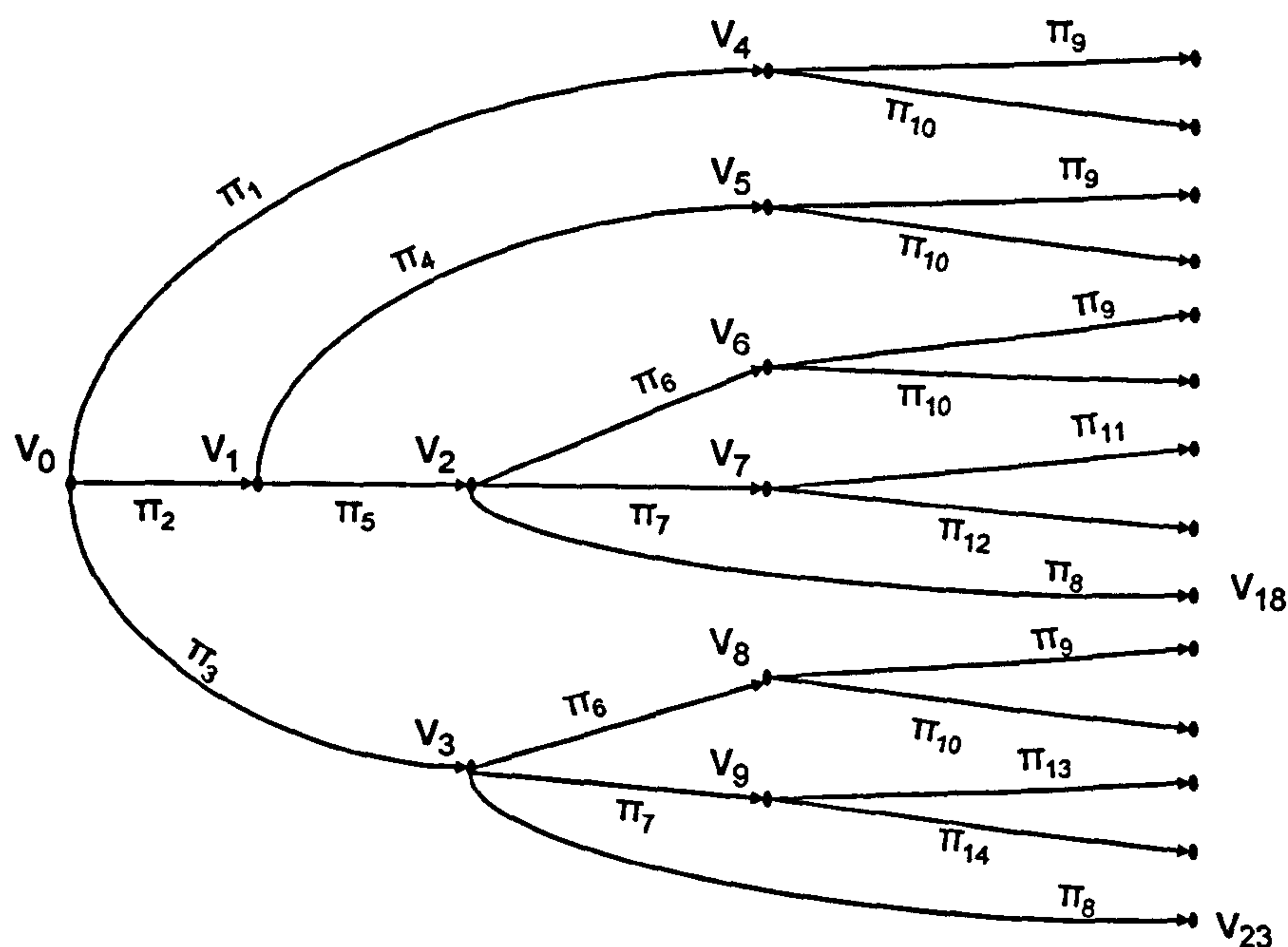


Figure 9: Tree for Machine Example

As before the process can be thought of as a series of four experiments with associated random variables (the 2nd, 3rd and 4th experiments may not happen).

In Example 3 we conscientiously followed the construction process outlined in section 2.4, but suggested how the process could be made considerably quicker by following the bullet points at the start of section 2.4. Here we simply follow these bullet points and hence do not need to define the variables X_i and Y_i for vertices v_0 to v_9 .

Step 1:

The edges leaving v_2 and v_3 carry identical probabilities, as do those leaving v_4, v_5, v_6 and v_8 , so we connect these sets of vertices with undirected edges, as in Figure 10.

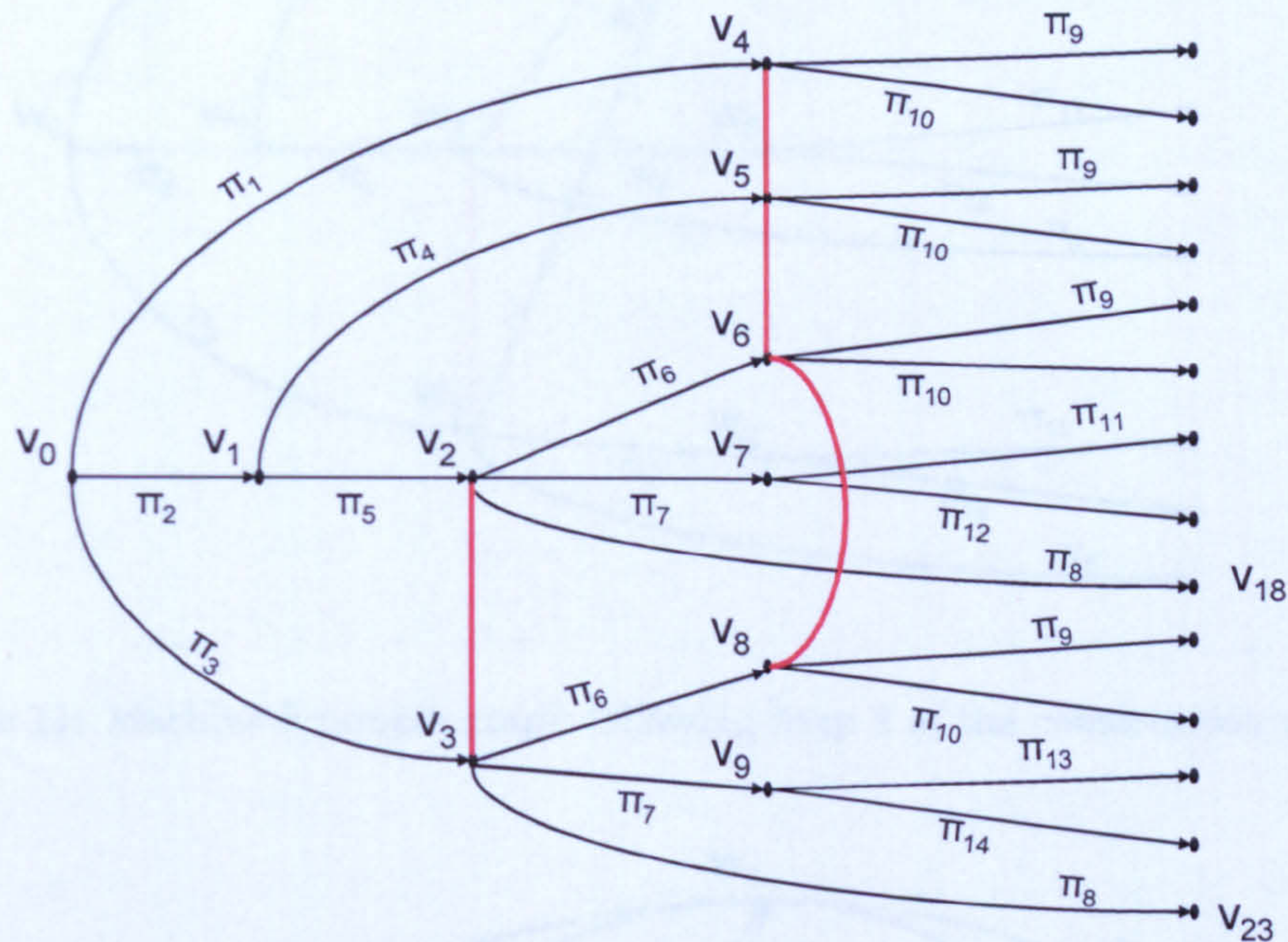


Figure 10: Augmented Tree for Machine Example

Step 2:

The subtrees rooted in the vertices v_4, v_5, v_6 and v_8 are identical both in topology and in the probabilities on their edges, so we combine these vertices into a single vertex (or position) w_4 . The resulting graph is shown in Figure 11.

Step 3:

Finally we combine all leaf-vertices into a single sink-vertex w_∞ , and add colour to edges as appropriate (colours are given by edge-labels from Figure 9). The final CEG is shown in Figure 12. Note that the vertices v_0 to v_9 have been replaced by positions w_0 to w_6 in such a way that if w_i precedes w_j on some path, then $i < j$. Note also that this labelling is consistent with that produced by following a partial ordering of equivalence classes as suggested in the construction process outlined in section 2.4.

Note how fast the construction process is in practice!

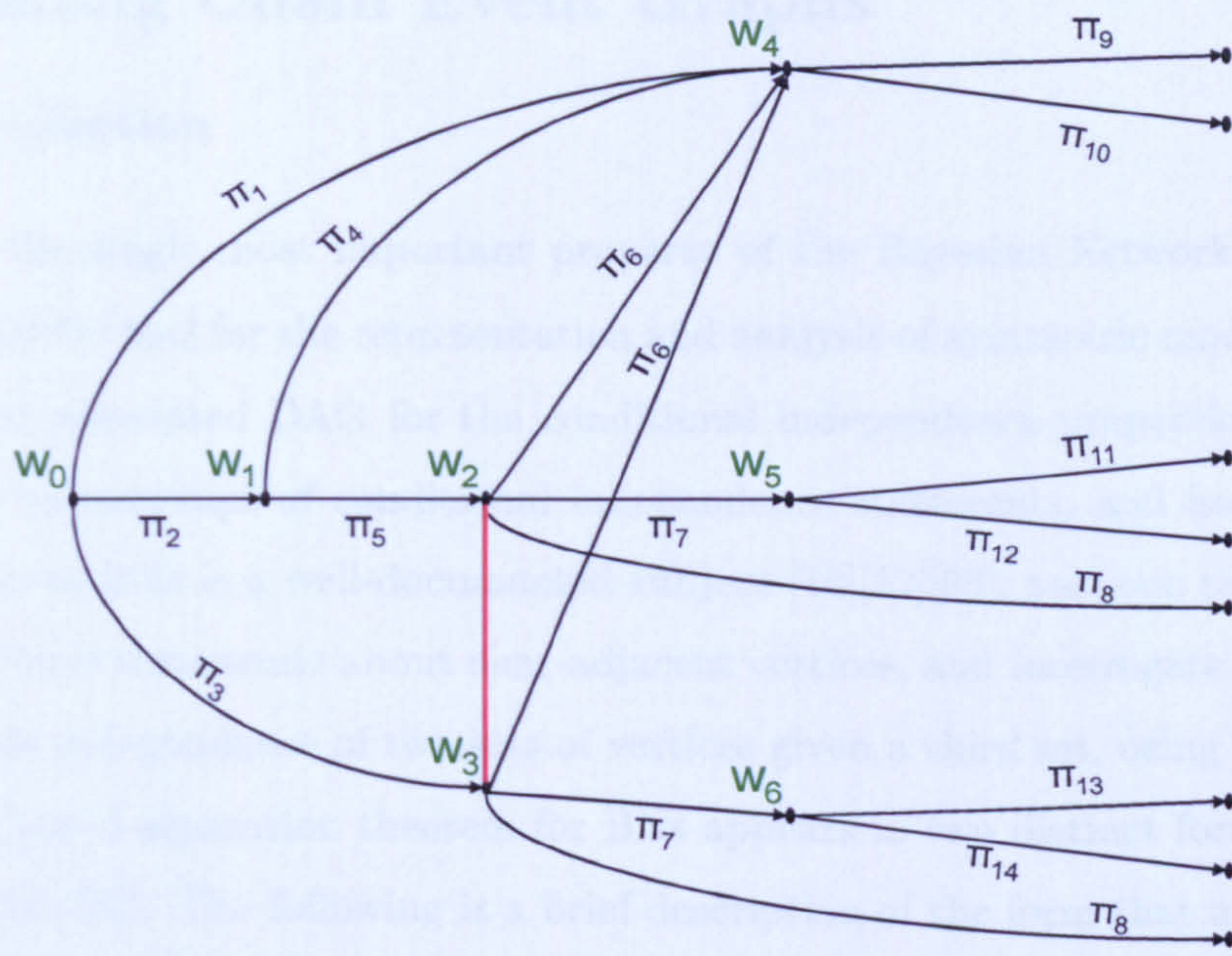


Figure 11: Machine Example graph following Step 2 of the construction process

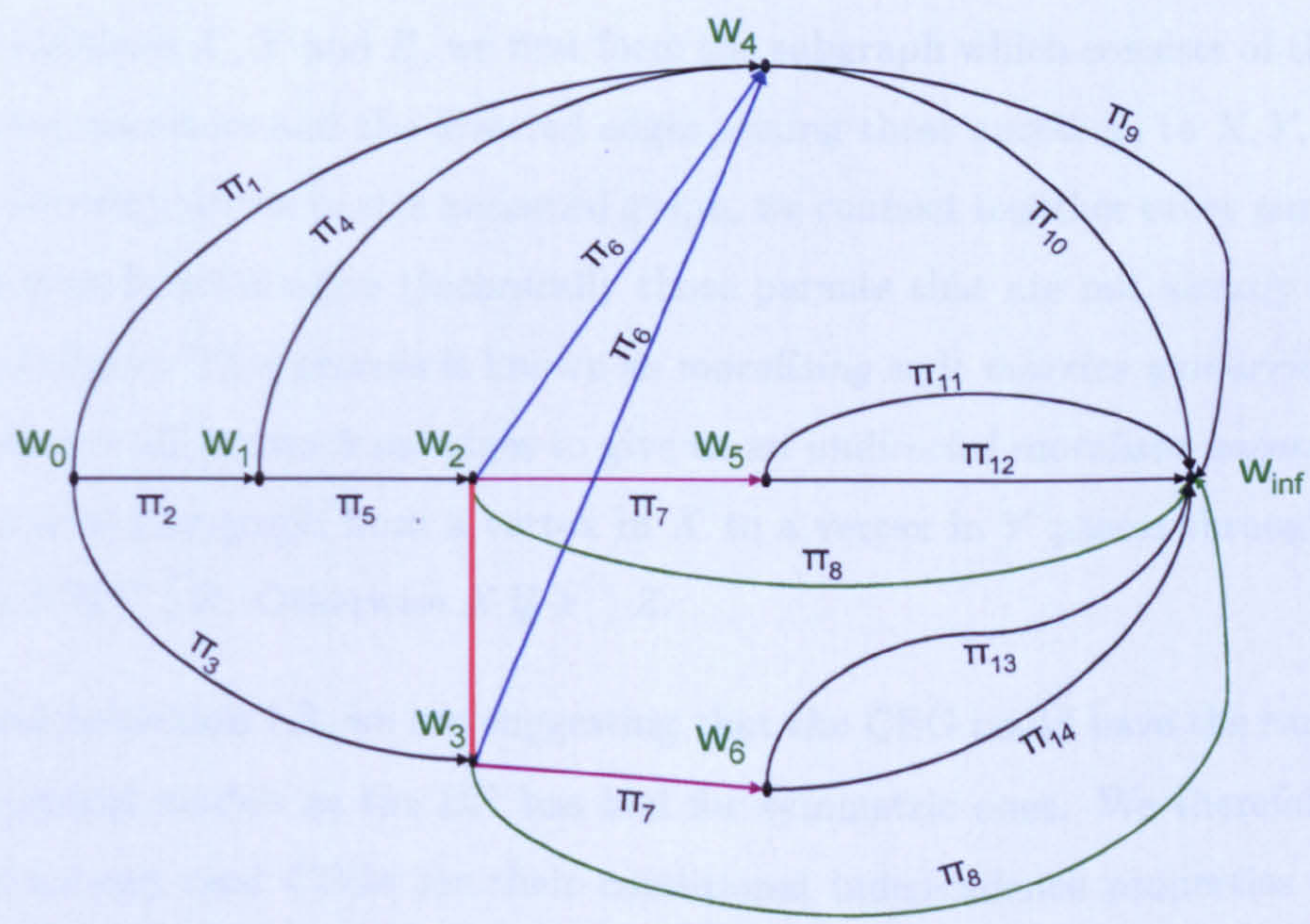


Figure 12: Final CEG for Machine Example

3. Reading Chain Event Graphs

3.1 Introduction

Probably the single most important property of the Bayesian Network which makes it such a powerful tool for the representation and analysis of symmetric models is our ability to read the associated DAG for the conditional independence properties of the system. The inter-relationships of conditional independence statements, and how they manifest themselves on BNs is a well-documented subject [10][52][58], and one can with practice, both read *local* statements about near-adjacent vertices, and interrogate the graph about the possible independence of two sets of vertices given a third set, using the d-separation theorem. The d-separation theorem for BNs appears in two distinct forms (described in detail in [41][30]). The following is a brief description of the form that appears in [30]:

If we consider a vertex V in a DAG, then the *parents* of V are the vertices from which there are directed edges into V , and the *ancestors* of V are the vertices from which there are directed paths leading to V . If we wish to interrogate the graph as to whether $X \perp\!\!\!\perp Y \mid Z$ for sets of variables X , Y and Z , we first form the subgraph which consists of the vertices X, Y, Z , their ancestors and the directed edges joining these ancestors to X, Y, Z .

Secondly, for every vertex in this ancestral graph, we connect together every parent of this vertex using undirected edges (technically those parents that are not already connected by directed edges). This process is known as *moralising* as it *marries unmarried* parents. We then remove all arrows from edges to give us an *undirected moralised ancestral* graph. If every path in this graph from a vertex in X to a vertex in Y passes through a vertex in Z , then $X \perp\!\!\!\perp Y \mid Z$. Otherwise $X \not\perp\!\!\!\perp Y \mid Z$.

As indicated in section 1.3, we are suggesting that the CEG could have the same impact for asymmetrical models as the BN has had for symmetric ones. We therefore need to show that we can read CEGs for their conditional independence properties as readily as one can read a BN. This means firstly that if we were to create a CEG for a totally symmetric model we would need to be able to show that any $X \perp\!\!\!\perp Y \mid Z$ statement readable from the equivalent BN (where X, Y, Z are sets of variables) is also readable from the CEG. Now, many asymmetric problems have conditional independence properties of the form $X \perp\!\!\!\perp Y \mid (Z = z)$ which only hold for specific (sets of) values of the variables Z . These cannot be represented on an unmodified BN. The context-specific BN [3] was created for this purpose, but even these are not really adequate for representing conditional independence properties of the form $X \perp\!\!\!\perp Y \mid \Lambda$ for some general event Λ .

So secondly, as our CEG is designed for use with asymmetric models, we will need to be able to read context-specific statements of the forms $X \perp\!\!\!\perp Y \mid (Z = z)$ and $X \perp\!\!\!\perp Y \mid \Lambda$ for some general event Λ , a union of the atomic root-to-sink paths of the CEG. In fact, not only can we read conditional independence properties such as these from a CEG, but we can also read statements of the forms $X \perp\!\!\!\perp \Lambda_2 \mid \Lambda_3$ or $\Lambda_1 \perp\!\!\!\perp \Lambda_2 \mid \Lambda_3$. These expressions reflect the CEG's event-based nature as opposed to the variable-based nature of the BN.

The derivation of the d-separation theorem for BNs was a slow process, and the equivalence of Pearl's theorem [41] and say Lauritzen's [30] is even now not readily apparent (a demonstration of this equivalence is given in [31]). Much progress has been made towards an equivalent d-separation theorem for CEGs, and this is presented in papers by J.Q. Smith [53][45] and also in this chapter where I present a new selection of powerful conditional-independence-statement types that can be read from a CEG. My emphasis in this thesis is on the primary importance of the $w_0 \rightarrow w_\infty$ paths of the CEG and event-based analysis. In particular in this chapter I am concerned with context-specific conditional independence which reflects the event-based topology of the CEG, so my approach to reading CEGs differs from that in [45][53]. This approach also leads to results that differ from those presented in these papers, which are concerned particularly with the idea of *cuts* of the CEG. A *cut* of a CEG is a collection of positions or stages such that every root-to-sink path in the CEG passes through a position within this collection. There are natural sets of conditional independence statements associated with such *cuts*. I expand a little on the idea of *cuts* in section 3.5, and give four examples of how they can help in the reading of CEGs.

It is worth noting at this point that we are not the first people to use the conditional independence properties of a model to attempt to modify the topology of a tree-representation of that model, with the hope of increasing the tree's usefulness as an analytical tool. Call and Miller in [4] (looking at Decision Trees) describe a process they call *coalescence*, whereby one can reduce tree size by taking advantage of the replication of subtrees. They spot that this property allows for some reading of the conditional independence structure, but they do not pursue this idea sufficiently to realise that there are essentially two distinct types of conditional independence within the tree, reflected in our equivalence (roughly corresponding to their coalescence), and our stage-equivalence. They note that the difficulties in reading conditional independence structure from trees has meant that analysts using trees have not fully taken advantage of the idea of coalescence.

They also, like French and Insua [15] compare the use of Decision Trees and Influence diagrams, and note that *the ability to exploit asymmetry can be a substantial advantage for trees. If trees could naturally exploit coalescence, the efficiency advantage is even greater.* I believe that in this thesis I demonstrate that the CEG is both the ideal representation for asymmetric problems, and also the ideal graph for the analysis of such problems. In this chapter I show how the topology of a CEG, and in particular its positions and stages, can be used to read the conditional independence properties of the model that it represents.

3.2 Basic ideas

This chapter contains the derivations and proofs of three significant results appertaining to the Reading of CEGs. The results themselves (given in section 3.3 and Proposition 5 in section 3.6) are easy to interpret and to use, but the proofs are rather long! Their significance (particularly that of Proposition 5) will become more evident in chapters 4 and 5 which are more concerned with the practicalities of analysis on CEGs, as opposed to the more theoretical apparatus discussed in this chapter. In this section I illustrate the basic ideas of the chapter through a running example. I start however with a useful result based on Definition 11 from chapter 2.

If a position in a CEG has more than one incoming edge (in fact if it has more than one incoming path), then it must be representative of an equivalence class of vertices in the original Tree which has more than one element. So, if in our Tree $W_k = \{v_i\}$ for $i \in I$, then:

$$\Lambda(w_k) = \bigcup_{i \in I} \Lambda(v_i)$$

where W_k is defined in Definition 9, $\Lambda(w)$ is defined in Definition 13, and $\Lambda(v)$ is defined in Definition 3 in chapter 2.

Now if $v_s, v_t \in \{v_i\}_{i \in I}$, then the Tree-based variables X_s and X_t (from Definition 4) are such that $X_s \equiv X_t$. Also, from Definition 11, the CEG-based variable X_k has the same sample space as X_s (X_t) and the same probability distribution over this space.

Proposition 2:

Suppose each of the vertices $v_i \in W_k$ and the position w_k has the criterion variable D say. Then:

$$\pi_{X_k}(d) = \pi(d \mid \Lambda(w_k))$$

for each outcome d of D .

Proof:

$$\Lambda(w_k) = \bigcup_{i \in I} \Lambda(v_i)$$

so:

$$\begin{aligned} \pi(d \mid \Lambda(w_k)) &= \pi(d \mid \bigcup_{i \in I} \Lambda(v_i)) \\ &= \frac{\pi(d, \bigcup_{i \in I} \Lambda(v_i))}{\pi(\bigcup_{i \in I} \Lambda(v_i))} \\ &= \frac{\sum_{i \in I} \pi(d, \Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda(v_i))} \end{aligned}$$

since the events $\{\Lambda(v_i)\}$ and $\{d, \Lambda(v_i)\}$ are disjoint

$$\begin{aligned} &= \frac{\sum_{i \in I} \pi(d \mid \Lambda(v_i)) \pi(\Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda(v_i))} \\ &= \frac{\sum_{i \in I} \pi_{X_i}(d) \pi(\Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda(v_i))} \end{aligned}$$

But $\pi_{X_s}(d) = \pi_{X_t}(d) \forall v_s, v_t \in \{v_i\}_{i \in I}$, so this equals:

$$= \frac{\sum_{i \in I} \pi_{X_s}(d) \pi(\Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda(v_i))}$$

for some specific $v_s \in W_k$.

$$\begin{aligned} &= \frac{\pi_{X_s}(d) \sum_{i \in I} \pi(\Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda(v_i))} \\ &= \pi_{X_s}(d) \\ &= \pi(d \mid \Lambda(v_s)) \quad \text{for } v_s \in \{v_i\}_{i \in I} \\ &= \pi_{X_k}(d) \end{aligned}$$

since X_k has the same sample space as X_s and the same probability distribution over this space.

Hence:

$$\pi_{X_k}(d) = \pi(d \mid \Lambda(w_k))$$

□

An exactly analogous result can be proved for the CEG-based variable Y_k :

$$\pi_{Y_k}(de \dots n) = \pi(de \dots n \mid \Lambda(w_k))$$

Example 1:

In this example we see how the conditional independence properties of a model are reflected in the topology of the CEG used to represent it. So, consider the Tree in Figure 1.

As in chapter 2 we can label our atomic events by $\lambda_1 = a_0 b_0 c_0 d_0$ etc., and we can define variables A, B, C such that:

$$\begin{aligned} a_0 &= \bigcup_{i=1}^6 \lambda_i & a_1 &= \bigcup_{i=7}^{12} \lambda_i \\ b_0 &= \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_7 \cup \lambda_8 \cup \lambda_9 \\ b_1 &= \lambda_4 \cup \lambda_5 \cup \lambda_6 \cup \lambda_{10} \cup \lambda_{11} \cup \lambda_{12} \\ c_0 &= \lambda_1 \cup \lambda_2 \cup \lambda_4 \cup \lambda_5 \cup \lambda_7 \cup \lambda_8 \cup \lambda_{10} \cup \lambda_{11} \\ c_1 &= \lambda_3 \cup \lambda_6 \cup \lambda_9 \cup \lambda_{12} \end{aligned}$$

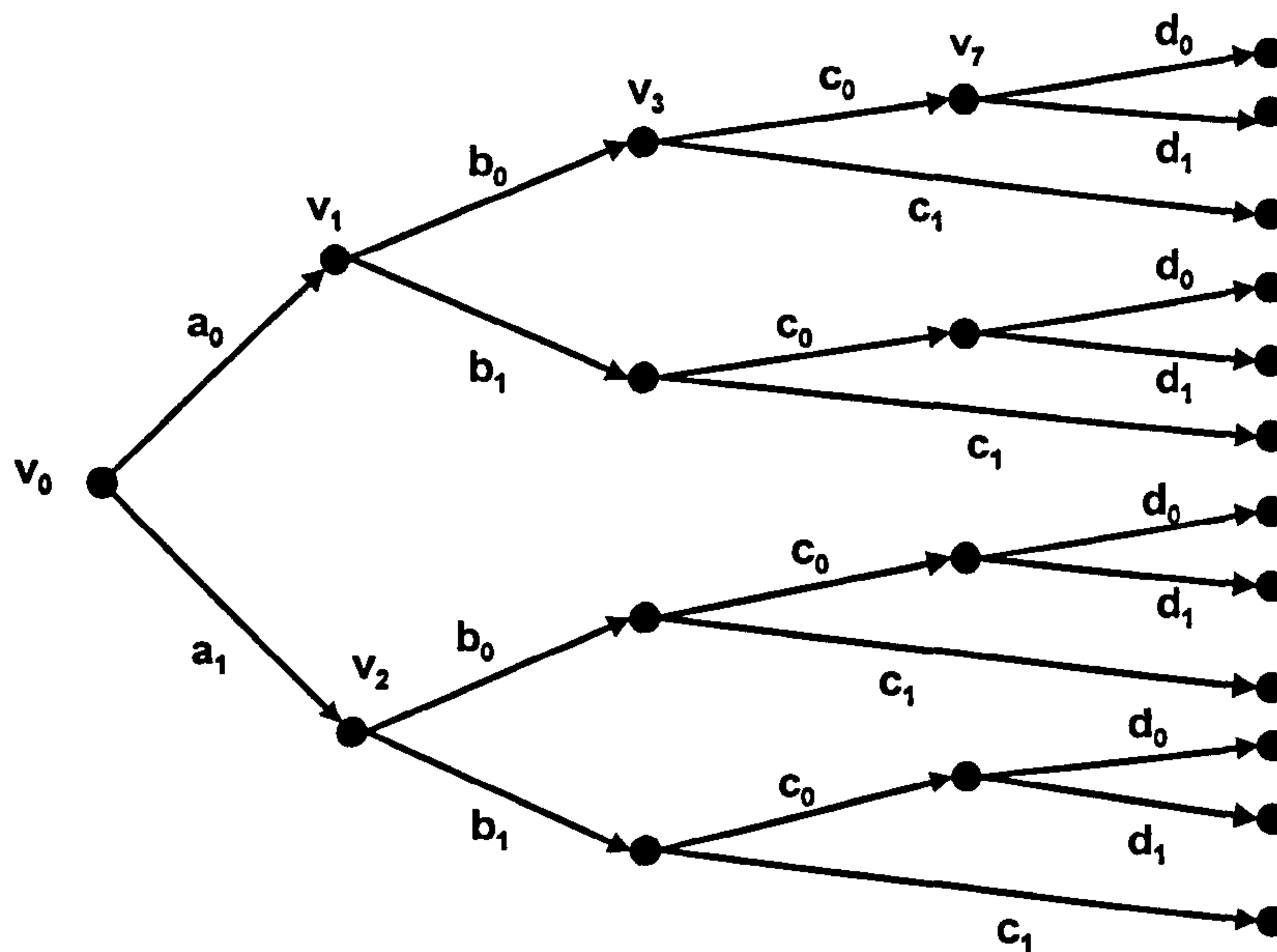


Figure 1: Tree for Example 1

We can also define the variable D taking values corresponding to the outcomes $\{d_0, d_1, d_\phi\}$, where d_0 is used to represent the event that is the union of all paths passing through an edge labelled d_0 ($= \lambda_1 \cup \lambda_4 \cup \lambda_7 \cup \lambda_{10}$), d_1 represents the event that is the union of all paths passing through an edge labelled d_1 ($= \lambda_2 \cup \lambda_5 \cup \lambda_8 \cup \lambda_{11}$), and d_ϕ represents the event that is the union of all paths **not** passing through an edge labelled d_0 or d_1 ($= \lambda_3 \cup \lambda_6 \cup \lambda_9 \cup \lambda_{12} = c_1$).

It is often stated that Trees do not illustrate any of the conditional independence structure of a model, but in fact for asymmetric models, some of this structure is actually apparent in the Tree. For example, we can see here that:

$$\pi(d_\phi \mid a_i b_j c_0) = 0 \quad \forall i, j$$

$$\pi(d_\phi \mid a_i b_j c_1) = 1 \quad \forall i, j$$

$$\text{and also that } \pi(d_i \mid a_j b_k c_1) = 0 \quad \text{for } i, j, k = 0, 1$$

These are of course context-specific conditional independence properties. Suppose we now impose some further conditional independence structure on our model, and require that:

$$\pi(c_i \mid a_0 b_j) = \pi(c_i \mid a_1 b_j) \quad \text{for } i, j = 0, 1$$

$$\pi(d_i \mid a_0 b_0 c_0) = \pi(d_i \mid a_1 b_0 c_0) \quad \text{for } i = 0, 1$$

Hence:

$$\pi(c_0 \mid a_0 b_0) = \pi(c_0 \mid a_1 b_0) = \pi(c_0 \mid b_0)$$

$$\pi(c_0 \mid a_0 b_1) = \pi(c_0 \mid a_1 b_1) = \pi(c_0 \mid b_1)$$

$$\pi(c_1 \mid a_0 b_0) = \pi(c_1 \mid a_1 b_0) = \pi(c_1 \mid b_0)$$

$$\pi(c_1 \mid a_0 b_1) = \pi(c_1 \mid a_1 b_1) = \pi(c_1 \mid b_1)$$

$$\pi(d_0 \mid a_0 b_0 c_0) = \pi(d_0 \mid a_1 b_0 c_0) = \pi(d_0 \mid b_0 c_0)$$

$$\pi(d_1 \mid a_0 b_0 c_0) = \pi(d_1 \mid a_1 b_0 c_0) = \pi(d_1 \mid b_0 c_0)$$

Figure 2 shows our Tree with all this information added, and Figure 3 our resultant CEG. An unmodified BN for this model would consist of 4 vertices A, B, C, D with directed edges joining A to B and D , B to C and D , and C to D . The only conditional independence property that could be read from this graph would be $C \perp\!\!\!\perp A \mid B$. The context-specific property that $D \perp\!\!\!\perp A \mid (b_0, C)$ (for example) cannot be read off the BN.

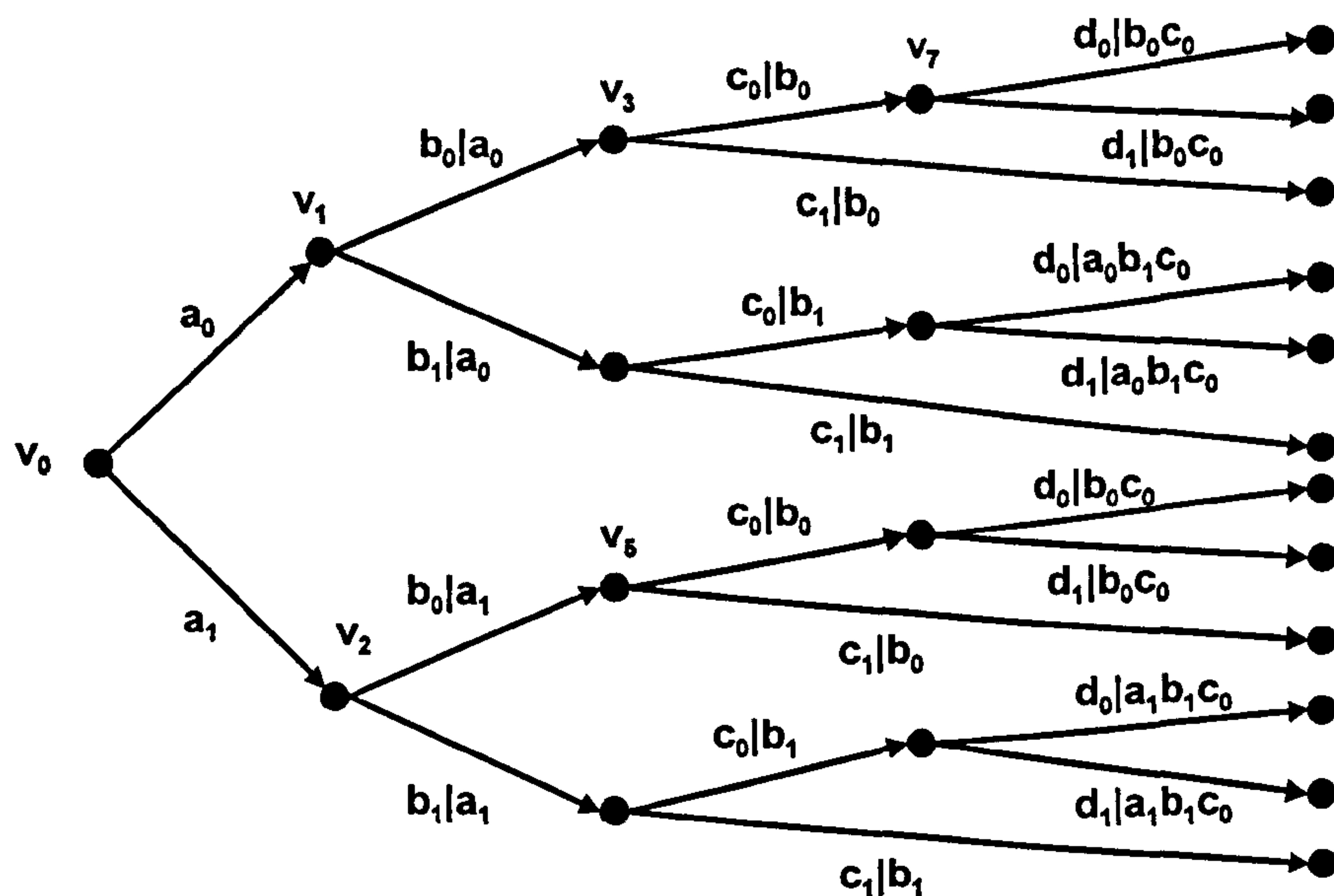


Figure 2: Example 1 Tree with probabilities added

Note that in our Tree, $W_3 = \{v_3, v_5\}$, so:

$$\begin{aligned}\Lambda(w_3) &= \Lambda(v_3) \cup \Lambda(v_5) \\ &= \{\lambda_1 \cup \lambda_2 \cup \lambda_3\} \cup \{\lambda_7 \cup \lambda_8 \cup \lambda_9\} \\ &= b_0\end{aligned}$$

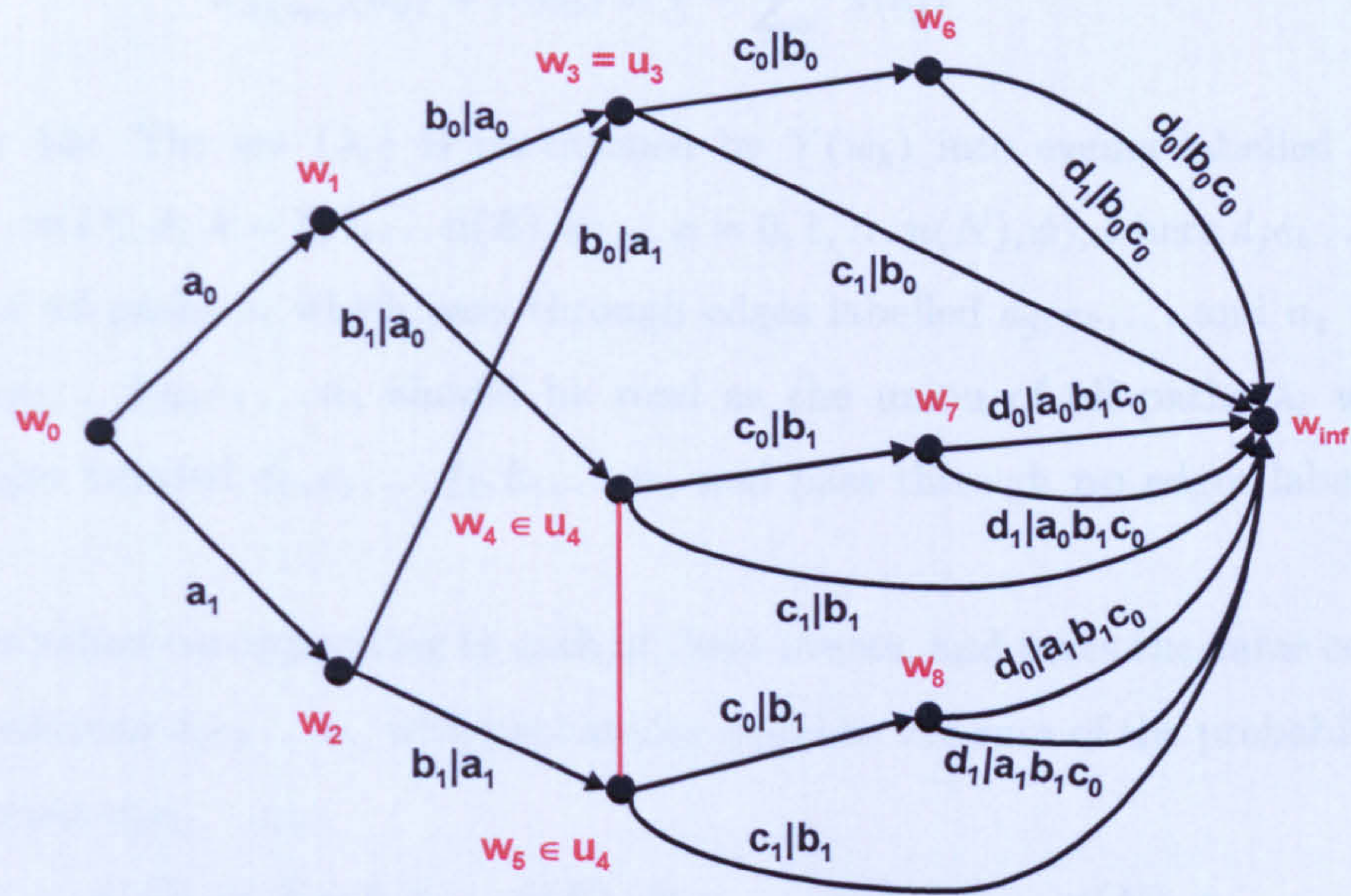


Figure 3: CEG for Example 1

We saw in chapter 2 that it was possible to define random variables on Trees and CEGs in a way that helped us to derive the CEG from a Tree. As we have seen, the variables $\{X_i\}$ and $\{Y_i\}$ from Definitions 10 and 11 have natural interpretations as variables associated with the

position w_i having the distribution of some conditional variable where the conditioning is upon $\Lambda(w_i)$. We now introduce some further random variables to help us in reading the CEG. These variables are similar in some ways to X_i and Y_i and have very straightforward natural interpretations.

Definition 14: The set $\{\lambda_i\}$ is partitioned by the variable $X(u_m)$ as follows:

Let the position w_k in our CEG correspond to a vertex v in our Tree, and let the vertex v have criterion variable D say (see section 2.3). Let positions w_k, w_l be such that $X_k \equiv X_l$ (see Definition 12), and let $w_k, w_l \in u_m$ for some stage u_m . Then the set $\{\lambda_i\}$ is partitioned by $X(u_m)$ into events labelled d_j ($j = 0, 1, \dots, n(D)$) and d_ϕ , where d_j labels the union of all paths λ_i which pass through an edge labelled d_j , and d_ϕ labels the union of all paths that pass through **no** edges labelled d_j for any j .

$X(u_m)$ takes values corresponding to the outcomes $\{d_0, d_1, \dots, d_{n(D)}, d_\phi\}$, and takes the value corresponding to the outcome d_j with probability equal to the sum of the probabilities of all λ_i in the subset d_j .

$$\pi_{X(u_m)}(d_j) = \pi(d_j) \quad \text{for } j = 0, 1, \dots, n(D)$$

$$\pi_{X(u_m)}(d_\phi) = \pi(d_\phi) = 1 - \sum_{j=0}^{n(D)} \pi(d_j)$$

Definition 15: The set $\{\lambda_i\}$ is partitioned by $Y(w_k)$ into events labelled $d_j e_k \dots n_z$ ($j = 0, 1, \dots, n(D), \phi$; $k = 0, 1, \dots, n(E), \phi$; $\dots z = 0, 1, \dots, n(N), \phi$), where $d_j e_k \dots n_z$ labels the union of all paths λ_i which pass through edges labelled d_j, e_k, \dots and n_z (where for example $d_1 e_1 \dots f_1 g_\phi h_1 \dots n_1$ should be read as the union of all paths λ_i which pass through edges labelled $d_1, e_1, \dots, f_1, h_1, \dots, n_1$ and pass through no edges labelled g_j for any j).

$Y(w_k)$ takes values corresponding to each of these events, and takes the value corresponding to the outcome $d_j e_k \dots n_z$ with probability equal to the sum of the probabilities of all λ_i in the subset $d_j e_k \dots n_z$.

For $j = 0, 1, \dots, n(D), \phi$; $k = 0, 1, \dots, n(E), \phi$; $\dots z = 0, 1, \dots, n(N), \phi$,

$$\pi_{Y(w_k)}(d_j e_k \dots n_z) = \pi(d_j e_k \dots n_z)$$

It could be argued that we have (through the addition of ϕ to the list of values taken by j, k, \dots, z) just defined the outcome space of $Y(w_k)$ as a Product Space. However, $Y(w_k)$ is a variable defined on the topology of the CEG – specifically it is associated with a particular position in the CEG and the sub-CEG rooted in that position, so we read each $d_j e_k \dots n_z$ here simply as a label for an event.

Definition 16: The set $\{\lambda_i\}$ is partitioned by $Z(w_k)$ into events labelled $a_j b_k \dots c_z$ ($j = 0, 1, \dots, n(A)$; $k = 0, 1, \dots, n(B), \phi$; $\dots z = 0, 1, \dots, n(C), \phi$).

$Z(w_k)$ takes values corresponding to each of these events, and takes the value corresponding to the outcome $a_j b_k \dots c_z$ with probability equal to the sum of the probabilities of all λ_i in the subset $a_j b_k \dots c_z$.

For $j = 0, 1, \dots, n(A)$; $k = 0, 1, \dots, n(B), \phi$; $\dots z = 0, 1, \dots, n(C), \phi$,

$$\pi_{Z(w_k)}(a_j b_k \dots c_z) = \pi(a_j b_k \dots c_z)$$

Definition 17: The set $\{\lambda_i\}$ is partitioned by $Z(u_m)$ into events labelled $a_j b_k \dots c_z$ ($j = 0, 1, \dots, n(A)$; $k = 0, 1, \dots, n(B), \phi$; $\dots z = 0, 1, \dots, n(C), \phi$).

$Z(u_m)$ takes values corresponding to each of these events, and takes the value corresponding to the outcome $a_j b_k \dots c_z$ with probability equal to the sum of the probabilities of all λ_i in the subset $a_j b_k \dots c_z$.

For $j = 0, 1, \dots, n(A)$; $k = 0, 1, \dots, n(B), \phi$; $\dots \dots z = 0, 1, \dots, n(C), \phi$,

$$\pi_{Z(u_m)}(a_j b_k \dots c_z) = \pi(a_j b_k \dots c_z)$$

Definition 18: For $u_m = \{w_j\}$, $j \in J$, define:

$$\Lambda(u_m) = \bigcup_{j \in J} \Lambda(w_j)$$

NB: As in section 2.3 these definitions assume that all root to sink paths have a consistent ordering.

3.3 Two important results

We can use these new variables to express two of the most fundamental properties of a CEG, which take the form of context-specific conditional independence statements. These statements, which are described for the first time here, are refinements of the concept of a cut described in [45][53] (and mentioned in section 3.1).

The first of these results can be expressed as follows:

Result 1: Any event defined on the outcome space of $X(u_m)$ is independent of any event defined on the outcome space of $Z(u_m)$ given the event $\Lambda(u_m)$.

So, for example, if the criterion variable for u_m is G then the outcome space of $X(u_m)$ is $\{g_0, g_1, \dots\}$, and events defined on the outcome space of $Z(u_m)$ will have labels that refer to the levels of a, b, c, d, e, f . An event defined on the outcome space of $Z(u_m)$ might therefore be labelled by $(a_0, b = d, f_2)$ for example. If now $\Lambda(u_m) = (b_0 c_0 f_2)$, then we get:

$$\begin{aligned} \pi(g_0 \mid (a_0, b = d, f_2), \Lambda(u_m)) &= \pi(g_0 \mid \Lambda(u_m)) \\ \pi(g_0 \mid (a_0, b = d, f_2), (b_0 c_0 f_2)) &= \pi(g_0 \mid a_0 b_0 c_0 d_0 f_2) \\ &= \pi(g_0 \mid b_0 c_0 f_2) \end{aligned}$$

Result 2: Any event defined on the outcome space of $Y(w_k)$ is independent of any event defined on the outcome space of $Z(w_k)$ given the event $\Lambda(w_k)$.

We can write these results as conditional independence expressions:

$$\begin{aligned} X(u_m) &\perp\!\!\!\perp Z(u_m) \mid \Lambda(u_m) \\ Y(w_k) &\perp\!\!\!\perp Z(w_k) \mid \Lambda(w_k) \end{aligned}$$

As the outcome space of $Y(w_k)$ represents the whole of the possible development of the CEG beyond w_k , then Result 2 tells us that we need nothing beyond the information contained in $\Lambda(w_k)$ in order to calculate the probability of any sub-path $w_k \rightarrow w_\infty$ in the outcome space of $Y(w_k)$. In this sense $\Lambda(w_k)$ can be thought of as the *history* of the position w_k , and indeed in [60] we called this event $q(w_k)$ ($Q(X)$ being typically used in BN-based analysis for variables which are *parents* of the variable X , and $q(x)$ being a value of $Q(X)$ evaluated at $X = x$).

Similarly, the outcome space of $X(u_m)$ represents the possible immediate development of the CEG beyond the positions $\{w\} \in u_m$, so Result 1 tells us that we need nothing beyond the information contained in $\Lambda(u_m)$ in order to calculate the probability on any edge leaving a position $w \in u_m$.

Clearly these two results are related to the d-separation theorem for BNs, as they allow us to use the topology of the CEG to establish elements of the conditional independence structure of the problem being represented. The results are proven in section 3.6, but I illustrate their importance below.

Definition 19: Analogously with Definitions 6, 8 and 12, if $X(u_m)$ and $X(u_n)$ have the same outcome space and the same probability distribution over this space, we write $X(u_m) \equiv X(u_n)$.

Definition 20: If $Y(w_k)$ and $Y(w_l)$ have the same outcome space and the same probability distribution over this space, we write $Y(w_k) \equiv Y(w_l)$.

Note that if w_k and w_l have the same criterion variable then $Y(w_k) \equiv Y(w_l)$ by Definition 15.

Example 2:

Consider the CEG from Figure 3 in section 3.2 (used to illustrate Example 1). Reading this, we see that $X(u_3) \equiv X(u_4)$, taking values corresponding to the outcomes $\{c_0, c_1\}$, with:

$$\pi_{X(u_3)}(c_i) = \pi_{X(u_4)}(c_i) = \pi(c_i)$$

Similarly $Y(w_3) \equiv Y(w_4) \equiv Y(w_5)$, taking values corresponding to the outcomes $\{c_0d_0, c_0d_1, c_1d_\phi\}$, with probabilities $\pi(c_0d_0)$, $\pi(c_0d_1)$, $\pi(c_1)$.

And $Z(w_3)$, $Z(w_4)$, $Z(w_5)$, $Z(u_3)$, $Z(u_4)$ take values corresponding to the outcomes $\{a_0b_0, a_0b_1, a_1b_0, a_1b_1\}$, with probabilities $\pi(a_0b_0)$, $\pi(a_0b_1)$, $\pi(a_1b_0)$, $\pi(a_1b_1)$.

$$\Lambda(w_3) = \Lambda(u_3) = b_0$$

$$\Lambda(w_4) = a_0 b_1$$

$$\Lambda(w_5) = a_1 b_1$$

$$\Lambda(u_4) = a_0 b_1 \cup a_1 b_1 = b_1$$

So $X(u_3) \amalg Z(u_3) \mid \Lambda(u_3)$ gives us:

$$\pi(c_i \mid (a_j b_k), b_0) = \pi(c_i \mid b_0)$$

$$\Rightarrow \pi(c_i \mid a_j b_0) = \pi(c_i \mid b_0)$$

$$\Rightarrow C \amalg A \mid b_0$$

as detailed below Figure 1 in section 3.2.

Similarly $X(u_4) \amalg Z(u_4) \mid \Lambda(u_4)$ gives us:

$$\pi(c_i \mid (a_j b_k), b_1) = \pi(c_i \mid b_1)$$

$$\Rightarrow \pi(c_i \mid a_j b_1) = \pi(c_i \mid b_1)$$

$$\Rightarrow C \amalg A \mid b_1$$

as also detailed in section 3.2.

Note that we can use the structure of the CEG in Figure 3 to write this as:

$$\pi(c_i \mid a_0 b_1) = \pi(c_i \mid a_1 b_1) = \pi(c_i \mid b_1)$$

$$\Rightarrow \pi(c_i \mid \Lambda(w_4)) = \pi(c_i \mid \Lambda(w_5)) = \pi(c_i \mid \Lambda(u_4))$$

This result can be generalised as follows:

Proposition 3:

Suppose each of the positions $w_k \in u_m$ and the stage u_m has the criterion variable D say.

Then:

$$\pi(d \mid \Lambda(u_m)) = \pi(d \mid \Lambda(w_k)) \quad \forall w_k \in u_m$$

for each outcome d of D .

Proof:

This proposition clearly has much in common with Proposition 2. Unsurprisingly the proof follows a similar path:

$$\Lambda(u_m) = \bigcup_k \Lambda(w_k) \quad \text{from Definition 18}$$

so:

$$\begin{aligned}
\pi(d \mid \Lambda(u_m)) &= \pi(d \mid \bigcup_k \Lambda(w_k)) \\
&= \frac{\pi(d, \bigcup_k \Lambda(w_k))}{\pi(\bigcup_k \Lambda(w_k))} \\
&= \frac{\sum_k \pi(d, \Lambda(w_k))}{\sum_k \pi(\Lambda(w_k))}
\end{aligned}$$

since the events $\{\Lambda(w_k)\}$ and the events $\{d, \Lambda(w_k)\}$ are disjoint

$$= \frac{\sum_k \pi(d \mid \Lambda(w_k)) \pi(\Lambda(w_k))}{\sum_k \pi(\Lambda(w_k))}$$

But $\pi(d \mid \Lambda(w_i)) = \pi(d \mid \Lambda(w_j))$ for all $w_i, w_j \in u_m$, so this equals:

$$= \frac{\sum_k \pi(d \mid \Lambda(w_1)) \pi(\Lambda(w_k))}{\sum_k \pi(\Lambda(w_k))}$$

for some specific $w_1 \in u_m$

$$\begin{aligned}
&= \frac{\pi(d \mid \Lambda(w_1)) \sum_k \pi(\Lambda(w_k))}{\sum_k \pi(\Lambda(w_k))} \\
&= \pi(d \mid \Lambda(w_1)) \\
&= \pi(d \mid \Lambda(w_k)) \quad \forall w_k \in u_m
\end{aligned}$$

□

Example 2 continued:

Note also that from the results for u_3 and u_4 we can here deduce that:

$$\pi(c_i \mid a_j b_k) = \pi(c_i \mid b_k)$$

and hence that in this CEG we have $C \amalg A \mid B$, a relationship between three variables.

Looking now at positions, we find that $Y(w_3) \amalg Z(w_3) \mid \Lambda(w_3)$ gives us:

$$\begin{aligned}
\pi(c_j d_k \mid (a_l b_m), b_0) &= \pi(c_j d_k \mid b_0) \\
\Rightarrow \pi(c_j d_k \mid a_l b_0) &= \pi(c_j d_k \mid b_0)
\end{aligned}$$

If we combine this result with that for u_3 (ie: $\pi(c_j \mid a_l b_0) = \pi(c_j \mid b_0)$), we get:

$$\begin{aligned}
\pi(d_k \mid a_l b_0 c_j) &= \frac{\pi(c_j d_k \mid a_l b_0)}{\pi(c_j \mid a_l b_0)} \\
&= \frac{\pi(c_j d_k \mid b_0)}{\pi(c_j \mid b_0)} \\
&= \pi(d_k \mid b_0 c_j) \\
\Rightarrow D \amalg A \mid (b_0, C)
\end{aligned}$$

a result completely consistent with those detailed beneath Figure 1 in section 3.2, but not (as already pointed out) readable from the unmodified BN of the problem.

3.4 Examples

Example 3: Another look at the Blood condition example

We now apply these ideas to some less artificial examples, and we start by looking at the Blood condition example from chapter 2. Figure 8 from section 2.6 is reproduced here to aid exposition.

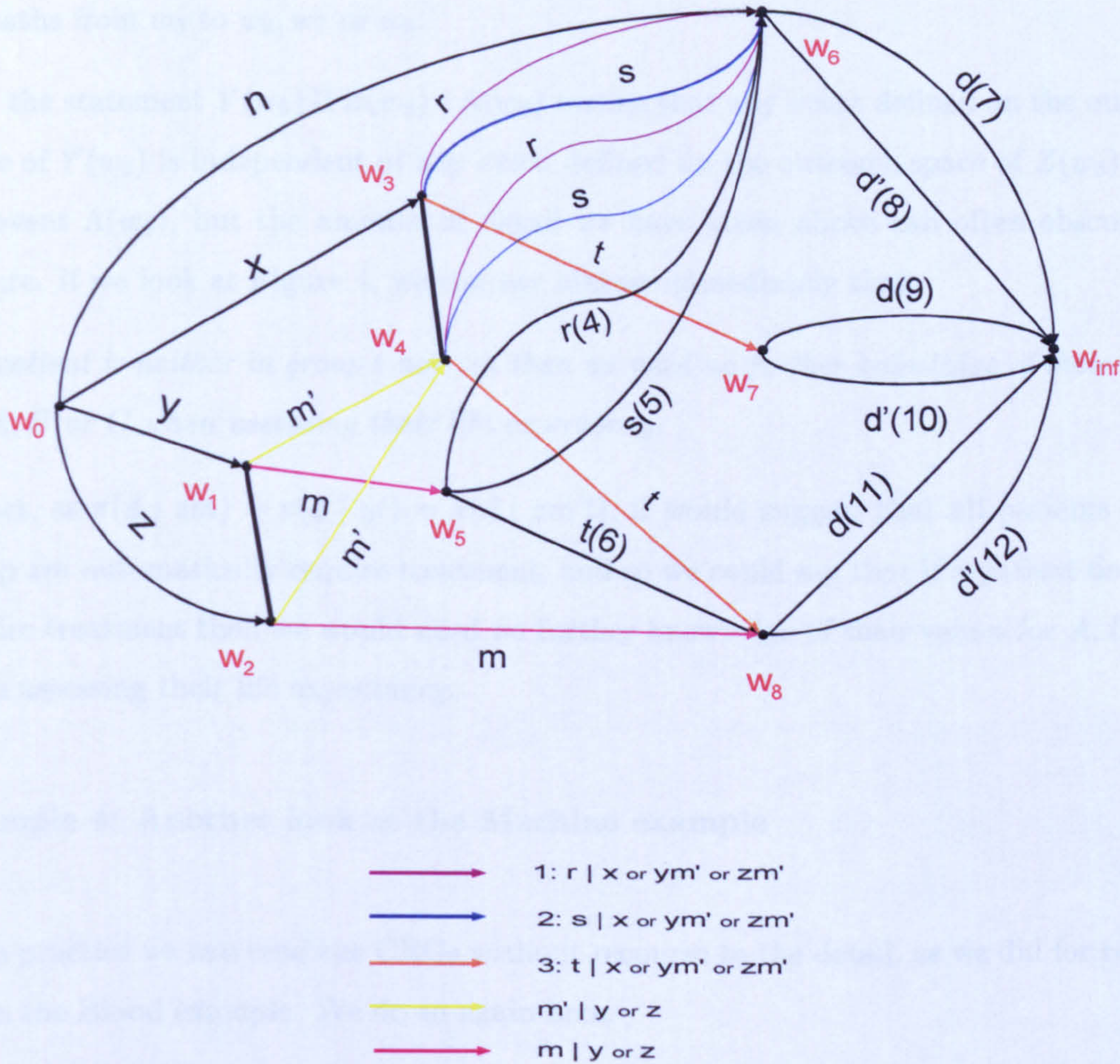


Figure 4: CEG for Blood Example

If we let $u_1 = \{w_1, w_2\}$, then we have $\Lambda(u_1) = y, z$.

$X(u_1)$ (which corresponds to B) takes values corresponding to the outcomes $\{m, m'\}$.

$Z(u_1)$ (which corresponds to A) takes values corresponding to the outcomes $\{n, x, y, z\}$.

We can read the statement $X(u_1) \amalg Z(u_1) \mid \Lambda(u_1)$ as:

$$\pi(m \mid y) = \pi(m \mid z) = \pi(m \mid y \text{ or } z)$$

$$\pi(m' \mid y) = \pi(m' \mid z) = \pi(m' \mid y \text{ or } z)$$

More interestingly, if we look at the position w_6 , we get:

$\Lambda(w_6)$ is the union of the events $n, xr, xs, yr, ys, zm'r, zm's$. These are the possible subpaths from w_0 to w_6 .

$Y(w_6)$ corresponds to D and takes values corresponding to the outcomes $\{d, d'\}$.

$Z(w_6)$ takes values corresponding to the outcomes $\{nb_\phi c_\phi, xb_\phi r, xb_\phi s, xb_\phi t, ymr, yms, ymt, ym'r, ym's, ym't, zmc_\phi, zm'r, zm's, zm't\}$. As the criterion variable for w_6 is D , these are the events labelled by values of A, B and C . They are the possible subpaths from w_0 to w_6, w_7 or w_8 .

Now the statement $Y(w_6) \perp\!\!\!\perp Z(w_6) \mid \Lambda(w_6)$ means that any event defined on the outcome space of $Y(w_6)$ is independent of any event defined on the outcome space of $Z(w_6)$ given the event $\Lambda(w_6)$, but the amount of detail we have given above can often obscure the picture. If we look at Figure 4, we can say almost immediately that:

If a patient is neither in group t nor zm then we need no further knowledge of their values for A, B or C when assessing their life expectancy.

In fact, as $\pi(d \mid zm) = \pi(d \mid yt) = \pi(d \mid zm't)$, it would suggest that all patients in the group zm automatically require treatment, and so we could say that if a patient does not require treatment then we would need no further knowledge of their values for A, B or C when assessing their life expectancy.

Example 4: Another look at the Machine example

With practice we can read the CEGs without recourse to the detail, as we did for reading w_6 in the Blood example. We do so again here.

Figure 5 below reproduces Figure 12 from chapter 2. If we let $u_2 = \{w_2, w_3\}$, then we have:

$\Lambda(u_2) = N$ indicated as faulty by monitoring process.

$X(u_2)$ takes values corresponding to the outcomes $\{N$ not actually faulty, N faulty and replaced successfully, N faulty and not replaced successfully $\}$.

$Z(u_2)$ takes values corresponding to the four possible combinations of M and N faulty/not faulty.

So the statement $X(u_2) \perp\!\!\!\perp Z(u_2) \mid \Lambda(u_2)$ tells us that knowledge of whether or not M was faulty is of no assistance in determining what happens to N once it has been indicated as faulty by the monitoring process.

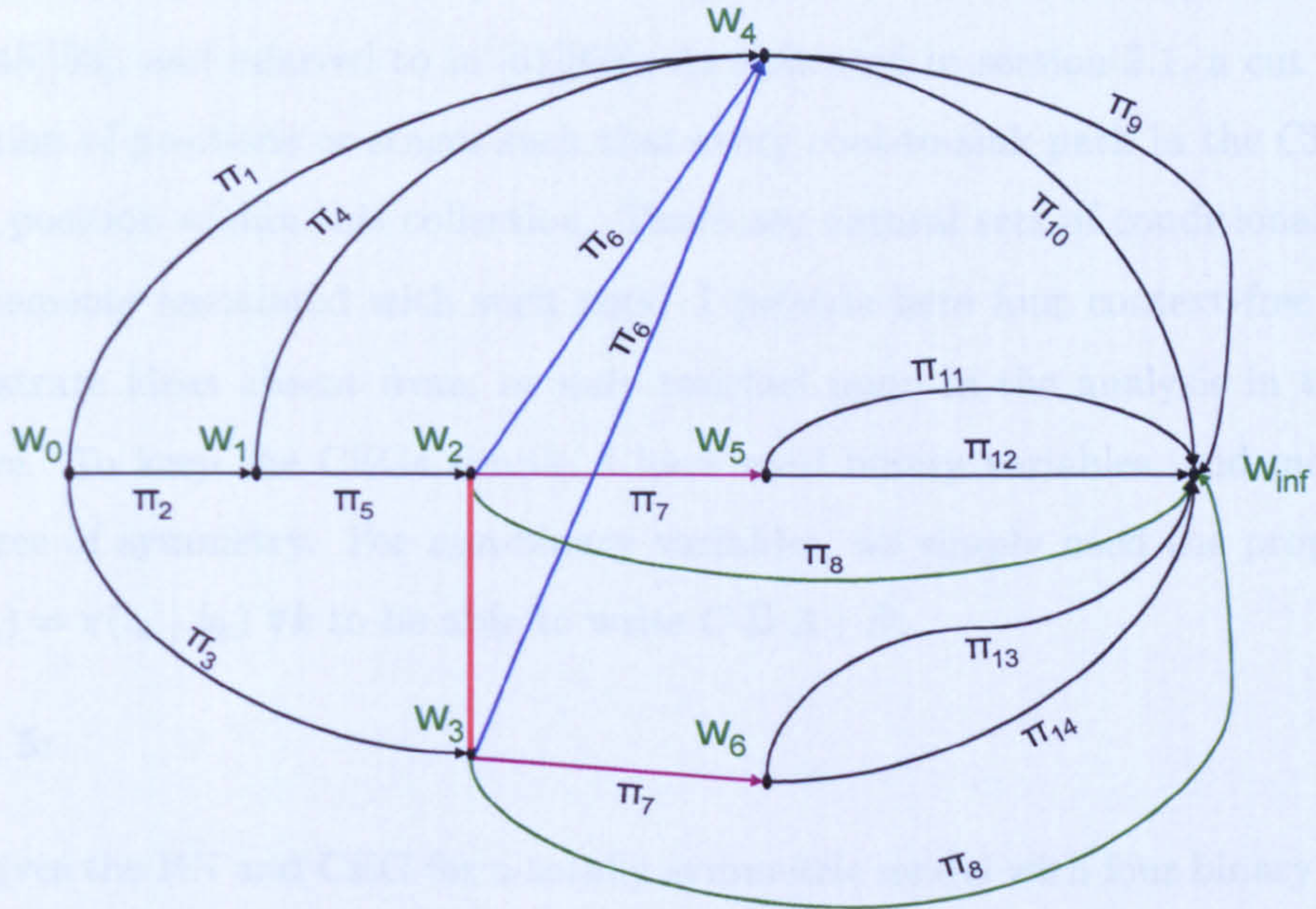


Figure 5: CEG for Machine Example

If we now consider the position w_4 , we get:

$\Lambda(w_4) = N$ is not **actually** faulty.

$Y(w_4)$ takes values corresponding to the outcomes {good product, bad product}.

$Z(w_4)$ takes values corresponding to all possible monitoring and replacement paths.

So the statement $Y(w_4) \amalg Z(w_4) \mid \Lambda(w_4)$ tells us that knowledge of the results of the **monitoring** process is of no assistance in determining product quality if we know that N is not **actually** faulty.

3.5 Cuts and Chain Event Graphs

In Example 2 in section 3.3 we found that for binary B , if:

$$\begin{aligned} \pi(c_i \mid a_j b_0) &= \pi(c_i \mid b_0) \\ \text{and } \pi(c_i \mid a_j b_1) &= \pi(c_i \mid b_1) \quad \forall i, j \end{aligned}$$

then we could write:

$$\begin{aligned} \pi(c_i \mid a_j b_k) &= \pi(c_i \mid b_k) \quad \forall k \\ \Rightarrow C \amalg A \mid B \end{aligned}$$

and hence a pair of conditional independence relationships between (variables and) events could be written as a single conditional independence relationship between variables. This

idea, which is related to the concept of a *cut* (described in section 3.1), is dealt with in detail in [45][53], and referred to in [61][60]. As indicated in section 3.1, a cut of a CEG is a collection of positions or stages such that every root-to-sink path in the CEG passes through a position within this collection. There are natural sets of conditional independence statements associated with such cuts. I provide here four context-free examples which illustrate ideas absent from, or only touched upon in the analysis in the papers cited above. To keep the CEGs simple, I have used binary variables, and models with a fair degree of symmetry. For non-binary variables, we simply need the property that $\pi(c_i | a_j b_k) = \pi(c_i | b_k) \forall k$ to be able to write $C \amalg A | B$.

Example 5:

Figure 6 gives the BN and CEG for a totally symmetric model with four binary variables.

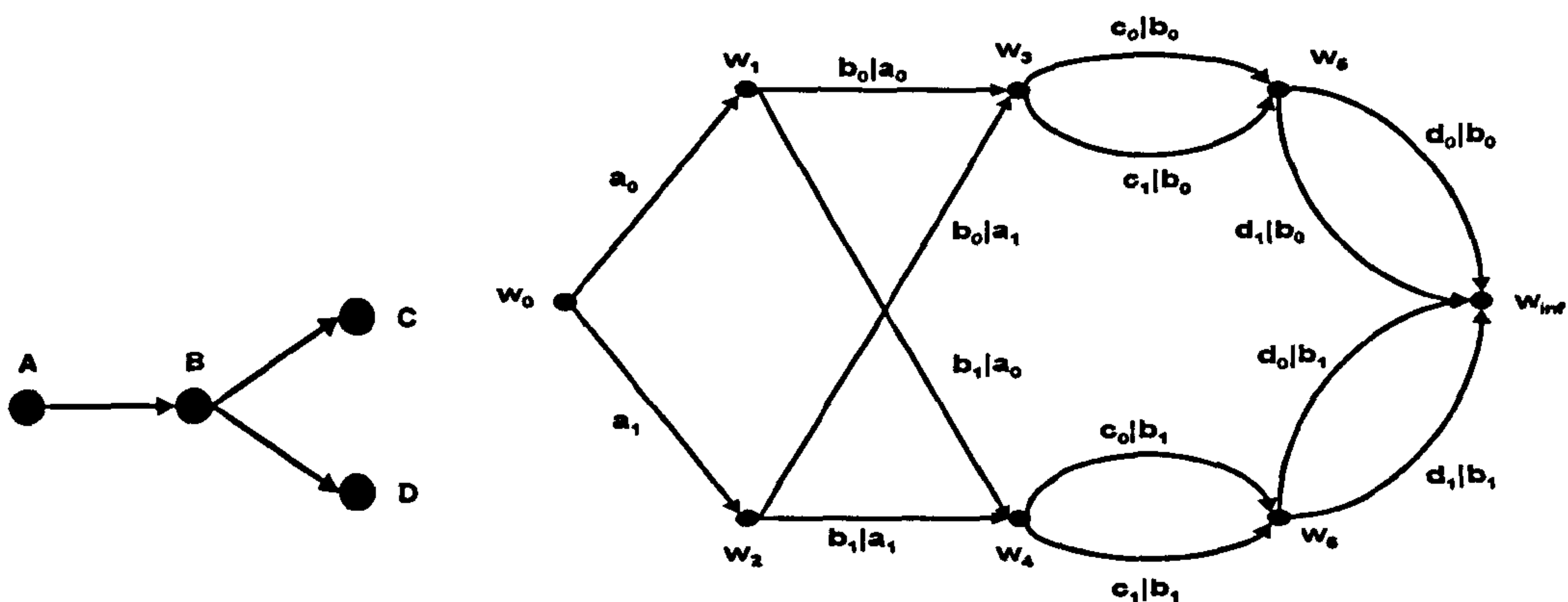


Figure 6: BN and CEG for Example 5

Looking at the CEG, we clearly have $\Lambda(w_3) = a_0, b_0 \cup a_1, b_0 = b_0$, so reading the position w_3 , we get:

$$\begin{aligned}
 & Y(w_3) \amalg Z(w_3) | \Lambda(w_3) \\
 & (C, D) \amalg (A, B) | b_0 \\
 \Rightarrow & (C, D) \amalg A | b_0 \\
 \Rightarrow & \pi(c_i d_j | a_k b_0) = \pi(c_i d_j | b_0) \quad \forall i, j, k
 \end{aligned}$$

It is often very useful to label positions in a more detailed fashion; and in simple near-symmetric CEGs there is a straightforward labelling process whereby we would label $w_3 = w(*, 0)$, where the $*$ indicates that the value taken by A is immaterial to $\Lambda(w_3)$, and the value taken by B corresponds to the outcome b_0 for all $w_0 \rightarrow w_3$ paths. Similarly, we would label $w_4 = w(*, 1)$.

A W -cut of C (Smith calls this a *fine cut* in [45][53]) is a collection W of positions $\{w\}$ such that every root-to-sink path in C passes through exactly one $w \in W$. A W -cut divides a CEG into upstream and downstream components.

So $\{w_3, w_4\}$ forms a W -cut. As B is binary, we can combine the statements:

$$\begin{aligned} \pi(c_i d_j \mid a_k b_0) &= \pi(c_i d_j \mid b_0) \\ \text{and } \pi(c_i d_j \mid a_k b_1) &= \pi(c_i d_j \mid b_1) \end{aligned}$$

into

$$\begin{aligned} \pi(c_i d_j \mid a_k b_l) &= \pi(c_i d_j \mid b_l) \quad \forall l \\ \Rightarrow (C, D) \amalg A \mid B \end{aligned}$$

a fairly typical sort of conditional independence statement produced by a W -cut. Similarly we can consider the W -cut $\{w_5, w_6\}$ to give $D \amalg (A, C) \mid B$. These are the two conditional independence properties we could read off the BN using the d-separation theorem.

Example 6:

Consider the BN and CEG in Figure 7. This is again a totally symmetric model with binary variables.

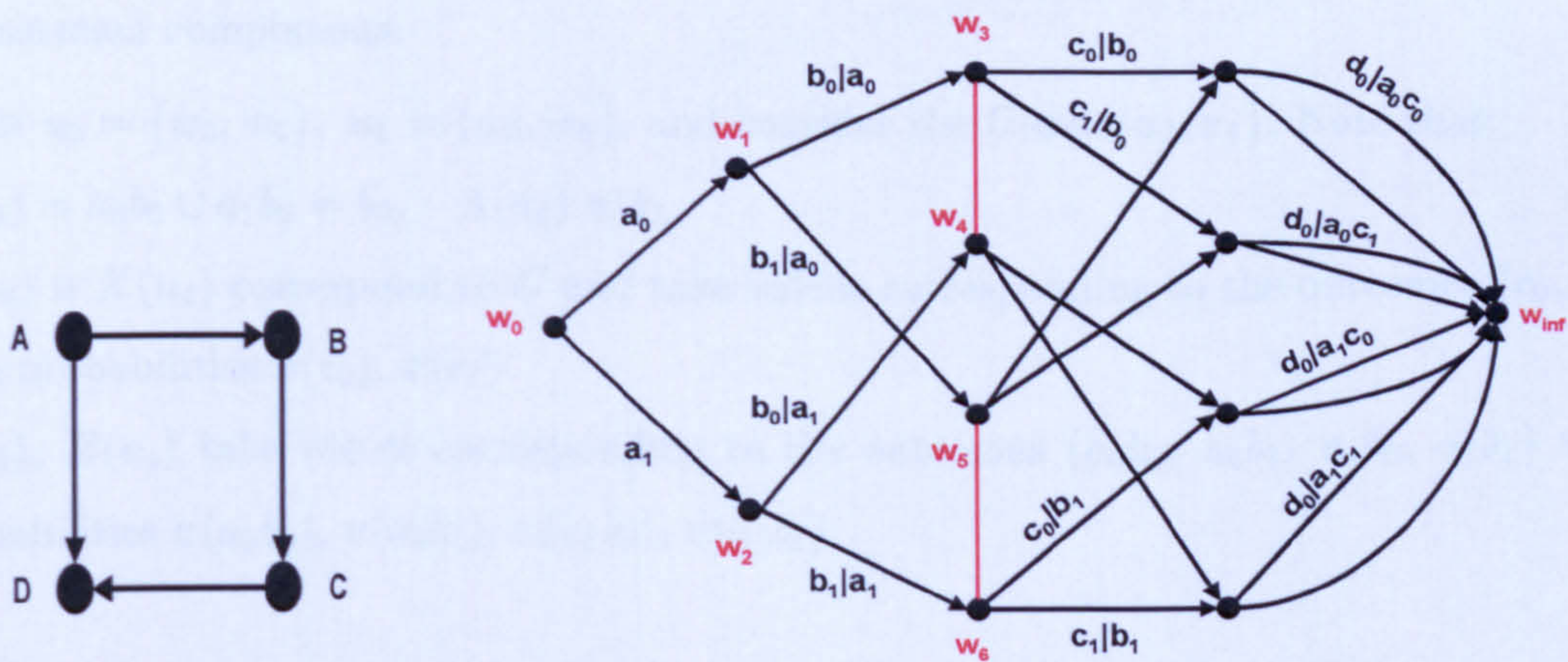


Figure 7: BN and CEG for Example 6

Before looking at the variables $Y(w), Z(w)$ etc., it is useful here to look back at the variables $\{Y_i\}$, since as they are closely related to the Tree-based construction variables, they can often tell us a lot about the conditional independence structure of the graph.

Consider the positions w_3 and w_4 . Note that w_3 and w_4 are **different** positions (with $\Lambda(w_3) = (a_0, b_0)$, $\Lambda(w_4) = (a_1, b_0)$), and hence we automatically have that:

$$Y_3 \neq Y_4$$

(Y_3 and Y_4 have the same outcome spaces as two different Tree based variables Y_i and Y_j , and the same probability distributions over these spaces. As the positions w_3 and w_4 are distinct then we cannot have $Y_i \equiv Y_j$ (see Definition 9), and hence $Y_3 \neq Y_4$)

So there must exist at least one pair (i, j) for which $\pi_{Y_3}(c_i d_j) \neq \pi_{Y_4}(c_i d_j)$

ie: there exists at least one pair (i, j) for which $\pi(c_i d_j \mid \Lambda(w_3)) \neq \pi(c_i d_j \mid \Lambda(w_4))$

ie: there exists at least one pair (i, j) for which $\pi(c_i d_j \mid a_0 b_0) \neq \pi(c_i d_j \mid a_1 b_0)$

$$\Rightarrow (C, D) \not\perp\!\!\!\perp A \mid b_0$$

$$\Rightarrow (C, D) \not\perp\!\!\!\perp A \mid B$$

whereas in Example 5, we had this statement as true. This fact may well be obvious from the BN, but the reading of context-specific dependencies (such as $(C, D) \not\perp\!\!\!\perp A \mid b_0$) is impossible from an unmodified BN used to represent an asymmetric problem. This BN may give us $(C, D) \not\perp\!\!\!\perp A \mid B$, but this doesn't always mean that $(C, D) \not\perp\!\!\!\perp A \mid b_0$.

In Example 5 we described a *W-cut*. Here we consider a *U-cut* (a *cut* in [45][53]). A *U-cut* of C is a collection U of stages $\{u\}$ such that every root-to-sink path in C passes through exactly one $w \in u$ for some $u \in U$. A *U-cut* divides a CEG into upstream and downstream components.

So let $u_3 = \{w_3, w_4\}$, $u_4 = \{w_5, w_6\}$, and consider the *U-cut* $\{u_3, u_4\}$. Note that:

$$\Lambda(u_3) = a_0 b_0 \cup a_1 b_0 = b_0, \quad \Lambda(u_4) = b_1$$

$X(u_3) \equiv X(u_4)$ correspond to C and take values corresponding to the outcomes $\{c_0, c_1\}$, with probabilities $\pi(c_0)$, $\pi(c_1)$

$Z(u_3)$, $Z(u_4)$ take values corresponding to the outcomes $\{a_0 b_0, a_0 b_1, a_1 b_0, a_1 b_1\}$ with probabilities $\pi(a_0 b_0)$, $\pi(a_0 b_1)$, $\pi(a_1 b_0)$, $\pi(a_1 b_1)$.

So:

$$X(u_3) \perp\!\!\!\perp Z(u_3) \mid \Lambda(u_3)$$

$$\Rightarrow C \perp\!\!\!\perp (A, B) \mid b_0$$

$$\Rightarrow C \perp\!\!\!\perp A \mid b_0$$

Similarly, looking at u_4 gives us:

$$C \perp\!\!\!\perp A \mid b_1$$

$$\Rightarrow C \perp\!\!\!\perp A \mid B$$

even though

$$(C, D) \not\perp\!\!\!\perp A \mid B$$

Note that a W -cut through the penultimate positions in the CEG gives us $D \perp\!\!\!\perp B \mid (A, C)$, which together with $C \perp\!\!\!\perp A \mid B$ are the conditional independence properties we could read off the BN using the d-separation theorem.

Example 7:

For the models in Examples 5 and 6 it is obviously quicker to use the BN than the CEG, but the CEG is principally designed for asymmetric models, so we now turn our attention to problems that cannot be adequately represented by BNs. Consider the CEG in Figure 8, and note that our four variables here all take the values 0 or 1.

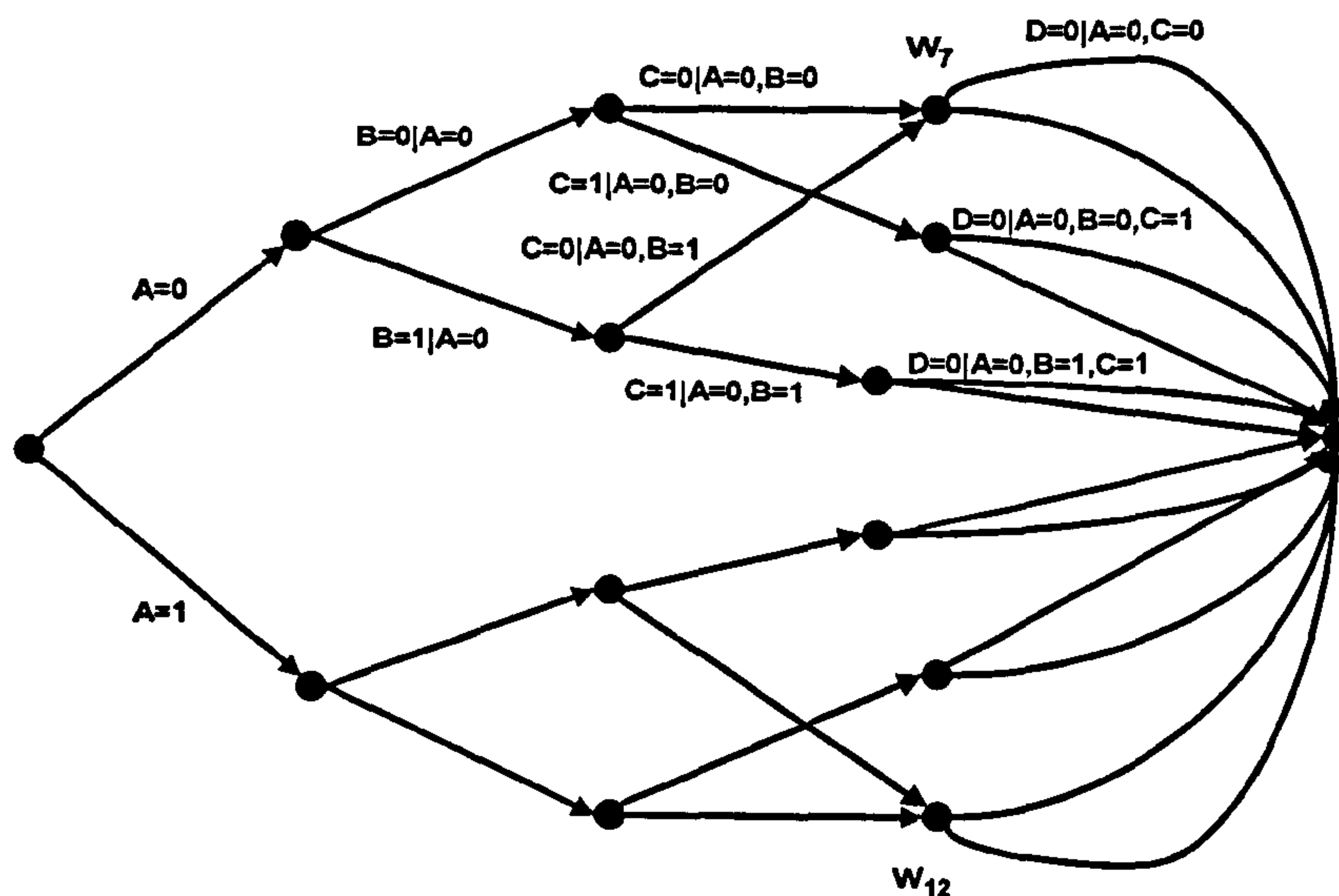


Figure 8: CEG for Example 7

The only two interesting positions here are w_7 and w_{12} . We have:

$$\Lambda(w_7) = (A = 0, C = 0) \quad (\text{hence we could label } w_7 \text{ as } w(0, *, 0))$$

$$\Lambda(w_{12}) = (A = 1, C = 1)$$

If we look at the W -cut through w_7, w_8, \dots, w_{12} , there is no obvious variable-based conditional independence statement that we can write, but we can see from w_7 and w_{12} that:

$$\begin{cases} Y(w_7) \perp\!\!\!\perp Z(w_7) \mid \Lambda(w_7) \\ Y(w_{12}) \perp\!\!\!\perp Z(w_{12}) \mid \Lambda(w_{12}) \end{cases}$$

$$\begin{aligned}
&\Rightarrow \begin{cases} D \amalg (A, B, C) \mid (A = 0, C = 0) \\ D \amalg (A, B, C) \mid (A = 1, C = 1) \end{cases} \\
&\Rightarrow \begin{cases} D \amalg B \mid (A = 0, C = 0) \\ D \amalg B \mid (A = 1, C = 1) \end{cases} \\
&\Rightarrow D \amalg B \mid A = C
\end{aligned}$$

which although expressed in terms of variables, is actually an event-based expression as $A = C$ represents an event ($= \lambda_1 \cup \lambda_2 \cup \lambda_5 \cup \lambda_6 \cup \lambda_{11} \cup \lambda_{12} \cup \lambda_{15} \cup \lambda_{16}$).

Note that this conditional independence statement would be very difficult to deduce from a Bayesian Network – the CEG in Figure 8 is only slightly asymmetric, but the model it represents cannot be adequately represented by a 4-vertex BN.

Example 8:

Consider the CEG in Figure 9, and note that our four variables again all take the values 0 or 1.

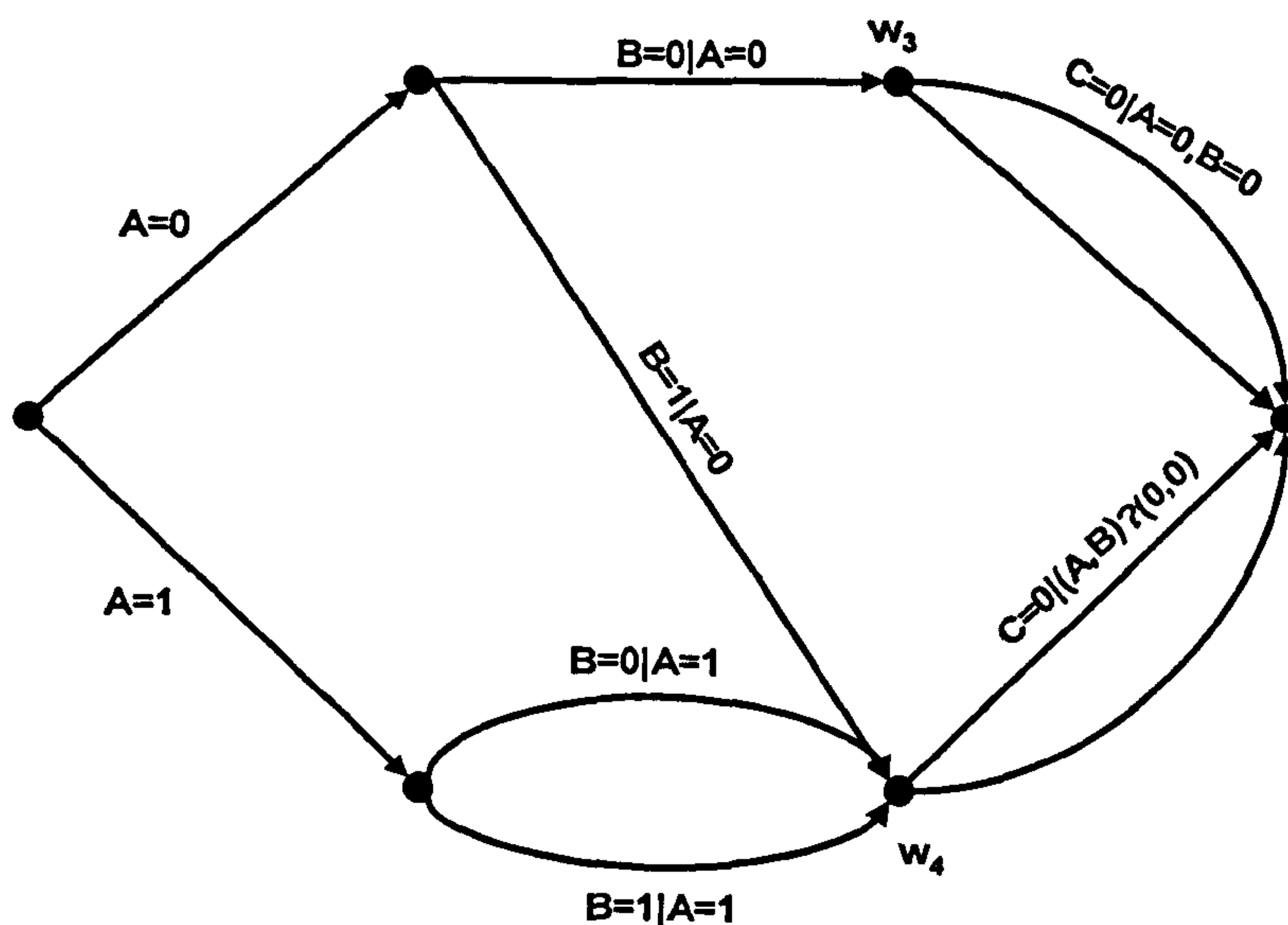


Figure 9: CEG for Example 8

Looking at w_3 and w_4 , we get:

$\Lambda(w_3) = (A = 0, B = 0)$ (so could label w_3 as $w(0, 0)$).

$\Lambda(w_4) = (A = 0, B = 1) \cup (A = 1, B = 0) \cup (A = 1, B = 1)$.

$Y(w_3) \equiv Y(w_4)$ correspond to C and take the values $\{C = 0, C = 1\}$.

$Z(w_3), Z(w_4)$ take values corresponding to the outcomes $\{A = 0, B = 0; A = 0, B = 1; A = 1, B = 0; A = 1, B = 1\}$.

So:

$$\begin{aligned} & \begin{cases} Y(w_3) \amalg Z(w_3) \mid \Lambda(w_3) \\ Y(w_4) \amalg Z(w_4) \mid \Lambda(w_4) \end{cases} \\ \Rightarrow & \begin{cases} C \amalg (A, B) \mid (A = 0, B = 0) \\ C \amalg (A, B) \mid ((A, B) \neq (0, 0)) \end{cases} \end{aligned}$$

which we could write as:

$$\begin{aligned} \Rightarrow & \begin{cases} C \amalg (A, B) \mid (Max(A, B) = 0) \\ C \amalg (A, B) \mid (Max(A, B) = 1) \end{cases} \\ \Rightarrow & C \amalg (A, B) \mid Max(A, B) \end{aligned}$$

This is actually a variable-based expression, as we could define a variable $D = Max(A, B)$:

$$\begin{cases} D = 0 & A = B = 0 \\ D = 1 & otherwise \end{cases}$$

D has a deterministic dependency on A and B . However, the model that the CEG in Figure 9 represents could not be adequately represented by either a 3-vertex or a 4-vertex BN, and this fact is reflected in the asymmetry of the CEG.

Note that the ideas introduced in this example can of course be developed for variables with more than two possible outcomes.

3.6 Semantics

This section deals with four important results first given in [61], but with perspective altered to reflect the emphasis on root-to-sink paths of this paper.

Throughout this section we will use the notation that $w_1 \sim w_2$ means that there is a directed edge joining the positions w_1 and w_2 ; and $w_1 \prec w_2$ means that there is a root-to-sink path that passes through both w_1 and w_2 , and w_1 precedes w_2 on this path. We will also use the shorthand $\pi(w_2 \mid w_1)$ to mean $\pi(\Lambda(w_2) \mid \Lambda(w_1))$.

Result 3:

For $w_1 \sim w_2$, $w_1 \prec w_2$, let $\pi_e(w_2 \mid w_1)$ be the probability labelling the (specified) edge $e(w_1, w_2)$. Note that there may be more than one edge joining w_1 and w_2 .

Then, following Definition 13 (4) from chapter 2, if we let $\Lambda(e(w_1, w_2))$ be the event that is the union of all $w_0 \rightarrow w_\infty$ paths that pass through the (specified) edge $e(w_1, w_2)$, we have:

$$\pi_e(w_2 \mid w_1) = \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1))$$

Example 9:

Consider the abbreviated CEG in Figure 10. Note that I am here using the position-labelling shorthand described in Example 5 in section 3.5.

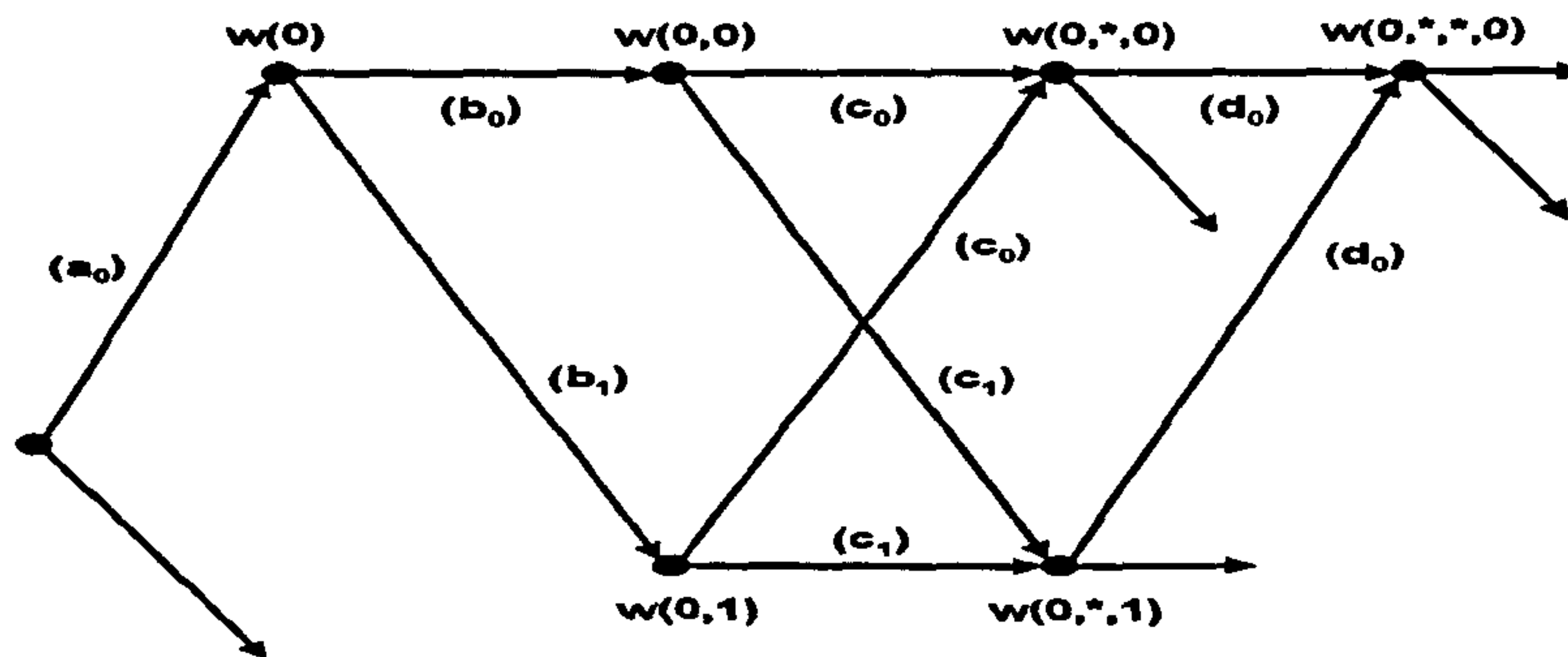


Figure 10: Abbreviated CEG for Examples 9, 10 and 11

Let $w_1 = w(0, *, 0)$ and $w_2 = w(0, *, *, 0)$. Then:

$$\Lambda(w_1) = a_0 c_0$$

$$\Lambda(w_2) = a_0 d_0$$

$$\Lambda(e(w_1, w_2)) = a_0 c_0 d_0$$

So the probability associated with the edge $e(w_1, w_2)$ is:

$$\begin{aligned} \pi_e(w_2 \mid w_1) &= \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1)) \\ &= \frac{\pi(\Lambda(w_1), \Lambda(e(w_1, w_2)))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(a_0 c_0 d_0)}{\pi(a_0 c_0)} \\ &= \pi(d_0 \mid a_0 c_0) \end{aligned}$$

Recall that each atomic event λ is a $w_0 \rightarrow w_\infty$ path, and that we use Λ for unions of such paths. It is also very useful to consider *path-segments* (sometimes referred to as *subpaths*), which bear the same relationship to $w_0 \rightarrow w_\infty$ paths as line-segments do to lines. So a path-segment is defined to be a specified set of edges joining two positions w_a, w_b say, where $w_a \prec w_b$ and either $w_0 \prec w_a$ or $w_b \prec w_\infty$ or both. We will use the symbol μ for such a path-segment: For $w_1 \prec w_2$, (w_1 not necessarily adjacent to w_2), let $\mu(w_1, w_2)$ be a (specified) path-segment from w_1 to w_2 .

Definition 21: For $w_1 \prec w_2$, (w_1 not necessarily adjacent to w_2), let $\Lambda(\mu(w_1, w_2))$ be the event that is the union of all $w_0 \rightarrow w_\infty$ paths that utilise the (specified) path-segment μ between w_1 and w_2 .

Result 4:

For $w_1 \prec w_2$, (w_1 not necessarily adjacent to w_2) and specified path-segment $\mu(w_1, w_2)$, let $\pi_\mu(w_2 \mid w_1)$ be the probability associated with the path-segment $\mu(w_1, w_2)$. Then:

$$\pi_\mu(w_2 \mid w_1) = \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))$$

Example 10:

In the abbreviated CEG given in Figure 10, let $w_1 = w(0)$, $w_2 = w(0, *, *, 0)$, and the specified path-segment $\mu(w_1, w_2)$ be the one comprising the edges joining $w(0)$ to $w(0, 1)$ to $w(0, *, 0)$ to $w(0, *, *, 0)$. Note that as in Example 9, we are considering a very simple situation where there is never more than one edge joining any two positions.

Then:

$$\Lambda(w_1) = a_0$$

$$\Lambda(w_2) = a_0 d_0$$

$$\Lambda(\mu(w_1, w_2)) = a_0 b_1 c_0 d_0$$

So the probability associated with the path-segment $\mu(w_1, w_2)$ is:

$$\begin{aligned} \pi_\mu(w_2 \mid w_1) &= \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1)) \\ &= \frac{\pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(a_0 b_1 c_0 d_0)}{\pi(a_0)} \\ &= \pi(b_1 c_0 d_0 \mid a_0) \end{aligned}$$

Definition 22: For $w_1 \prec w_2$, let $\Lambda(w_1, w_2)$ be the event that is the union of all $w_0 \rightarrow w_\infty$ paths that pass through both w_1 and w_2 .

Note that $\Lambda(w_1, w_2) = \Lambda(w_1) \cap \Lambda(w_2)$. This result is obvious if we think of $\Lambda(w)$ as a union of atoms: If $\Lambda(w_1) = \cup_i \lambda_i$, $\Lambda(w_2) = \cup_j \lambda_j$, then $\Lambda(w_1, w_2) = (\cup_i \lambda_i) \cap (\cup_j \lambda_j) = \Lambda(w_1) \cap \Lambda(w_2)$.

Proposition 4:

For $w_1 \prec w_2$, with $M = \{\mu(w_1, w_2)\}$, the set of all path-segments connecting w_1 and w_2 :

$$\pi(w_2 \mid w_1) = \sum_{\mu \in M} \pi_\mu(w_2 \mid w_1)$$

Proof:

$$\begin{aligned}\pi(w_2 \mid w_1) &= \pi(\Lambda(w_2) \mid \Lambda(w_1)) \\ &= \pi(\Lambda(w_1, w_2) \mid \Lambda(w_1))\end{aligned}$$

But we can write $\Lambda(w_1, w_2) = \cup_{\mu \in M} \Lambda(\mu(w_1, w_2))$, so:

$$\begin{aligned}\pi(w_2 \mid w_1) &= \pi\left(\bigcup_{\mu \in M} \Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1)\right) \\ &= \frac{\pi(\bigcup_{\mu \in M} (\Lambda(w_1), \Lambda(\mu(w_1, w_2))))}{\pi(\Lambda(w_1))} \\ &= \frac{\sum_{\mu \in M} \pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)))}{\pi(\Lambda(w_1))}\end{aligned}$$

since the events $\{\Lambda(w_1), \Lambda(\mu(w_1, w_2))\}$ are disjoint

$$\begin{aligned}&= \sum_{\mu \in M} \left[\frac{\pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)))}{\pi(\Lambda(w_1))} \right] \\ &= \sum_{\mu \in M} \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1)) \\ &= \sum_{\mu \in M} \pi_{\mu}(w_2 \mid w_1)\end{aligned}$$

□

Example 11:

In the abbreviated CEG given in Figure 10, let $w_1 = w(0)$, $w_2 = w(0, *, *, 0)$. Then:

$$\begin{aligned}\Lambda(w_1) &= a_0 \\ \Lambda(w_2) &= a_0 d_0 \\ \Lambda(w_1, w_2) &= \Lambda(w_1) \cap \Lambda(w_2) = a_0 d_0\end{aligned}$$

and:

$$\begin{aligned}\pi(w_2 \mid w_1) &= \pi(\Lambda(w_1, w_2) \mid \Lambda(w_1)) \\ &= \frac{\pi(\Lambda(w_1), \Lambda(w_2))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(a_0 d_0)}{\pi(a_0)} \\ &= \pi(d_0 \mid a_0)\end{aligned}$$

Or from first principles:

Let $\mu_1(w_1, w_2)$ be the path-segment between the positions $w(0)$ and $w(0, *, *, 0)$ which consists of the edges labelled b_0, c_0, d_0 ; $\mu_2(w_1, w_2)$ be the path-segment between the positions $w(0)$ and $w(0, *, *, 0)$ which consists of the edges labelled b_0, c_1, d_0 ; $\mu_3(w_1, w_2)$ be

the path-segment between the positions $w(0)$ and $w(0, *, *, 0)$ which consists of the edges labelled b_1, c_0, d_0 ; and $\mu_4(w_1, w_2)$ be the path-segment between the positions $w(0)$ and $w(0, *, *, 0)$ which consists of the edges labelled b_1, c_1, d_0 .

So:

$$\Lambda(\mu_1(w_1, w_2)) = a_0 b_0 c_0 d_0$$

$$\Lambda(\mu_2(w_1, w_2)) = a_0 b_0 c_1 d_0$$

$$\Lambda(\mu_3(w_1, w_2)) = a_0 b_1 c_0 d_0$$

$$\Lambda(\mu_4(w_1, w_2)) = a_0 b_1 c_1 d_0$$

and:

$$\begin{aligned} \pi(w_2 \mid w_1) &= \sum_{\mu \in M} \pi_\mu(w_2 \mid w_1) \\ &= \sum_{\mu \in M} \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1)) \\ &= \sum_{b, c} \pi(a_0 b c d_0 \mid a_0) \\ &= \pi(d_0 \mid a_0) \end{aligned}$$

Definition 23: For $w_1 \prec w_2 \prec w_3$, let:

$$\pi(w_3 \mid w_1, w_2) \triangleq \pi(\Lambda(w_3) \mid \Lambda(w_1, w_2))$$

We are now in a position to express another fundamental property of the CEG, which will be invaluable in proofs of propositions in both chapters 4 and 5. As with Results 1 and 2, this property is described for the first time in this thesis. Proposition 5 below verifies for CEGs a well-known result for Trees, which however has a somewhat more significant meaning in the context of a CEG.

Proposition 5:

For $w_1 \prec w_2 \prec w_3$:

$$\pi(w_3 \mid w_1, w_2) = \pi(w_3 \mid w_2)$$

I provide two proofs to this proposition. The first is possibly more mathematically pleasing as it respects the path sigma-algebra of the CEG by not splitting atoms. It is however, very long, and rather subtle, being based on the topology of the Tree from which the CEG is derived.

Proof of Proposition 5:

Throughout this proof we use the word *path* not to mean an atomic root-to-leaf path, but simply to mean a *route* along edges. When we wish to refer to a root-to-leaf path this will be made explicit.

In our CEG consider positions w_1, w_2, w_3 such that $w_1 \prec w_2 \prec w_3$. We know that the position w_i ($i = 1, 2, 3$) in our CEG is representative of an equivalence class W_i of vertices in our original Tree. So let:

$$W_1 = \{v_1\}$$

$$W_2 = \{v_2\} = \{v_2^j\} \quad \text{if individual vertices need to be identified}$$

$$W_3 = \{v_3\}$$

Now, because the subtrees rooted in each v_1 are identical, there will be at least one path from each v_1 of the form $v_1 \prec v_2 \prec v_3$. Similarly, since the subtrees rooted in each v_2 are identical, there will be at least one path from each v_2 of the form $v_2 \prec v_3$. In fact, if there are m distinct paths from w_2 to w_3 in our CEG, there will be m distinct paths of the form $v_2^j \prec v_3^{jk}$ ($k = 1, \dots, m$) for each $v_2^j \in W_2$. Note however that there may be $w_0 \prec w_2$ paths not passing through w_1 , so there may exist $v_2 \in W_2$ for which there exist no $v_1 \in W_1$ such that $v_1 \prec v_2$.

Let $W_2^0 = \{v_2 \in W_2 \mid \exists v_1 \in W_1 \text{ such that } v_1 \prec v_2\}$.

Since the subtrees rooted in each v_2 are identical, it is logical to index those $v_3 \in W_3$ such that $v_2 \prec v_3$ for some v_2 , in the same order for each v_2^j . Then the *corresponding* path-segments from each v_2^j to v_3^{jk} ($k = 1, \dots, m$) will have the same probabilities for each $v_2^j \in W_2$. We write:

$$\begin{aligned} \pi_k &= \pi(\Lambda(v_3^{jk}) \mid \Lambda(v_2^j)) \\ &= \pi(\Lambda(v_3^k) \mid \Lambda(v_2)) \end{aligned}$$

a constant not dependent on which $v_2 \in W_2$ is being considered.

For (specified) $v_1 \in W_1, v_2 \in W_2^0, v_3 \in W_3$ such that $v_1 \prec v_2 \prec v_3$, consider the event $\Lambda(v_1, v_2, v_3)$ which is the union of all root-to-leaf paths passing through v_1, v_2 and v_3 , and which has the appearance of a path-segment $v_0 \prec v_1 \prec v_2 \prec v_3$ conjoined to the subtree rooted in v_3 .

But in a Tree there is only ever one path from v_0 to any other vertex, so the path $v_0 \prec v_1 \prec v_2 \prec v_3$ is the only path $v_0 \prec v_3$. Moreover, for specified v_1, v_2, v_3 , we have that:

$$\Lambda(v_1, v_2, v_3) = \Lambda(v_1, v_3) = \Lambda(v_2, v_3) = \Lambda(v_3) \quad (3.6.1)$$

Similarly, for specified $v_1 \in W_1, v_2 \in W_2^0$ such that $v_1 \prec v_2$, we have that:

$$\Lambda(v_1, v_2) = \Lambda(v_2) \quad (3.6.2)$$

Now for $w_1 \prec w_2$:

$$\Lambda(w_1, w_2) = \bigcup_{v_1 \in W_1, v_2 \in W_2} \Lambda(v_1, v_2)$$

so:

$$\begin{aligned} \pi(\Lambda(w_1, w_2)) &= \pi\left(\bigcup_{v_1 \in W_1, v_2 \in W_2} \Lambda(v_1, v_2)\right) \\ &= \sum_{v_1 \in W_1, v_2 \in W_2} \pi(\Lambda(v_1, v_2)) \end{aligned}$$

since the events $\{\Lambda(v_1, v_2)\}$ are disjoint

$$= \sum_{v_2 \in W_2^0, v_1 \prec v_2} \pi(\Lambda(v_1, v_2))$$

since $\pi(\Lambda(v_1, v_2)) = 0$ for any $v_2 \notin W_2^0$, and any $v_1 \not\prec v_2 \in W_2^0$

$$= \sum_{v_2 \in W_2^0} \pi(\Lambda(v_2))$$

using (3.6.2), and noting that for each $v_2 \in W_2^0$ there is only one $v_1 \in W_1$ such that $v_1 \prec v_2$.

Similarly, for $w_1 \prec w_2 \prec w_3$:

$$\Lambda(w_1, w_2, w_3) = \bigcup_{v_1 \in W_1, v_2 \in W_2, v_3 \in W_3} \Lambda(v_1, v_2, v_3)$$

so:

$$\begin{aligned} \pi(\Lambda(w_1, w_2, w_3)) &= \pi\left(\bigcup_{v_1 \in W_1, v_2 \in W_2, v_3 \in W_3} \Lambda(v_1, v_2, v_3)\right) \\ &= \sum_{v_1 \in W_1, v_2 \in W_2, v_3 \in W_3} \pi(\Lambda(v_1, v_2, v_3)) \end{aligned}$$

since the events $\{\Lambda(v_1, v_2, v_3)\}$ are disjoint

$$= \sum_{v_2 \in W_2^0, v_1 \prec v_2, v_3 \succ v_2} \pi(\Lambda(v_1, v_2, v_3))$$

since $\pi(\Lambda(v_1, v_2, v_3)) = 0$ for any $v_2 \notin W_2^0$, and any $v_1 \not\prec v_2 \in W_2^0$, and any $v_3 \not\succ v_2 \in W_2^0$

$$= \sum_{v_2 \in W_2^0, v_3 \succ v_2} \pi(\Lambda(v_2, v_3))$$

using (3.6.1) above.

$$\begin{aligned} &= \sum_{v_2^j \in W_2^0} \sum_{k=1}^m \pi(\Lambda(v_2^j, v_3^{jk})) \\ &= \sum_{v_2^j \in W_2^0} \sum_{k=1}^m \pi(\Lambda(v_3^{jk}) \mid \Lambda(v_2^j)) \pi(\Lambda(v_2^j)) \\ &= \sum_{v_2^j \in W_2^0} \sum_{k=1}^m \pi_k \pi(\Lambda(v_2^j)) \end{aligned}$$

where π_k is not dependent on which $v_2^j \in W_2^0$ is being considered

$$= \sum_{k=1}^m \pi_k \sum_{v_2^j \in W_2^0} \pi(\Lambda(v_2^j))$$

Substituting into the expression for $\pi(w_3 \mid w_1, w_2)$, we get:

$$\begin{aligned} \pi(w_3 \mid w_1, w_2) &= \pi(\Lambda(w_3) \mid \Lambda(w_1, w_2)) \\ &= \frac{\pi(\Lambda(w_1, w_2), \Lambda(w_3))}{\pi(\Lambda(w_1, w_2))} \\ &= \frac{\pi(\Lambda(w_1, w_2, w_3))}{\pi(\Lambda(w_1, w_2))} \\ &= \frac{\sum_{k=1}^m \pi_k \sum_{v_2^j \in W_2^0} \pi(\Lambda(v_2^j))}{\sum_{v_2^j \in W_2^0} \pi(\Lambda(v_2^j))} \\ &= \sum_{k=1}^m \pi_k \end{aligned}$$

Now, also:

$$\Lambda(w_2) = \bigcup_{v_2 \in W_2} \Lambda(v_2)$$

so:

$$\begin{aligned} \pi(\Lambda(w_2)) &= \pi\left(\bigcup_{v_2 \in W_2} \Lambda(v_2)\right) \\ &= \sum_{v_2 \in W_2} \pi(\Lambda(v_2)) \end{aligned}$$

since the events $\{\Lambda(v_2)\}$ are disjoint.

And for $w_2 \prec w_3$:

$$\Lambda(w_2, w_3) = \bigcup_{v_2 \in W_2, v_3 \in W_3} \Lambda(v_2, v_3)$$

so:

$$\begin{aligned}\pi(\Lambda(w_2, w_3)) &= \pi\left(\bigcup_{v_2 \in W_2, v_3 \in W_3} \Lambda(v_2, v_3)\right) \\ &= \sum_{v_2 \in W_2, v_3 \in W_3} \pi(\Lambda(v_2, v_3))\end{aligned}$$

since the events $\{\Lambda(v_2, v_3)\}$ are disjoint

$$= \sum_{v_2 \in W_2, v_3 \succ v_2} \pi(\Lambda(v_2, v_3))$$

since $\pi(\Lambda(v_2, v_3)) = 0$ for any $v_3 \not\succ v_2$

$$\begin{aligned}&= \sum_{v_2^j \in W_2} \sum_{k=1}^m \pi(\Lambda(v_2^j, v_3^{jk})) \\ &= \sum_{v_2^j \in W_2} \sum_{k=1}^m \pi(\Lambda(v_3^{jk}) \mid \Lambda(v_2^j)) \pi(\Lambda(v_2^j)) \\ &= \sum_{v_2^j \in W_2} \sum_{k=1}^m \pi_k \pi(\Lambda(v_2^j))\end{aligned}$$

where π_k is **not** dependent on which $v_2^j \in W_2$ is being considered

$$= \sum_{k=1}^m \pi_k \sum_{v_2^j \in W_2} \pi(\Lambda(v_2^j))$$

Substituting into the expression for $\pi(w_3 \mid w_2)$, we get:

$$\begin{aligned}\pi(w_3 \mid w_2) &= \pi(\Lambda(w_3) \mid \Lambda(w_2)) \\ &= \frac{\pi(\Lambda(w_2), \Lambda(w_3))}{\pi(\Lambda(w_2))} \\ &= \frac{\pi(\Lambda(w_2, w_3))}{\pi(\Lambda(w_2))} \\ &= \frac{\sum_{k=1}^m \pi_k \sum_{v_2^j \in W_2} \pi(\Lambda(v_2^j))}{\sum_{v_2^j \in W_2} \pi(\Lambda(v_2^j))} \\ &= \sum_{k=1}^m \pi_k\end{aligned}$$

Hence:

$$\pi(w_3 \mid w_1, w_2) = \pi(w_3 \mid w_2)$$

□

a result which is trivial for Trees, but obviously not so for CEGs! It is worth noting that whilst a CEG is an elegant way of representing a problem and an ideal structure for the topological checking of conditional independence statements, it is often worth thinking of them as equivalence classes of Trees if we wish to prove fundamental results.

My second proof to Proposition 5 is much shorter and relies on the idea that a root-to-sink path in a CEG can be expressed as the conjunction of a set of subpaths. Before I proceed to the proof I discuss in more detail the terms $\pi_e(w_2 \mid w_1)$ and $\pi_\mu(w_2 \mid w_1)$ introduced in Results 3 and 4. $\Lambda(e(w_1, w_2))$ is the event that is the union of all $w_0 \rightarrow w_\infty$ paths that pass through the edge $e(w_1, w_2)$. That is, it is the event that one passes along the edge $e(w_1, w_2)$ on one's progress from w_0 to w_∞ . The probability on this edge (ie. the probability used to label the edge) is the probability that one passes along the edge given that one passes through the position w_1 ; that is:

$$\pi_e(w_2 \mid w_1) = \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1))$$

As we have defined $\pi_\mu(w_4 \mid w_3)$ analogously, this is clearly the probability of utilising the subpath $\mu(w_3, w_4)$ given that one passes through the position w_3 , and we can express $\pi_\mu(w_4 \mid w_3)$ as the product of terms $\pi_e(w_j \mid w_i)$ for the edges $\{e(w_i, w_j)\}$ comprising the subpath $\mu(w_3, w_4)$.

So consider now a $w_0 \rightarrow w_\infty$ path:

$$\lambda = \lambda(w_0, e_\alpha, w_1, e_\beta, w_2, e_\gamma, w_3, e_\delta, w_\infty)$$

We can express:

$$\pi(\lambda) = \pi_\lambda(w_\infty \mid w_0) = \pi(\lambda(w_0, w_\infty) \mid \Lambda(w_0))$$

as the product of the probabilities on the edges $e_\alpha, e_\beta, e_\gamma, e_\delta$. That is:

$$\pi(\lambda) = \pi_{e_\alpha}(w_1 \mid w_0) \pi_{e_\beta}(w_2 \mid w_1) \pi_{e_\gamma}(w_3 \mid w_2) \pi_{e_\delta}(w_\infty \mid w_3)$$

Suppose now that λ consists of 2 conjoined subpaths:

$$\mu_a = \mu_a(w_0, e_\alpha, w_1, e_\beta, w_2) \quad \text{and}$$

$$\mu_b = \mu_b(w_2, e_\gamma, w_3, e_\delta, w_\infty)$$

Then the probability of getting from w_0 to w_2 via μ_a is the probability of utilising the subpath $\mu_a(w_0, w_2)$ given we start at w_0 , which is $\pi_{\mu_a}(w_2 \mid w_0)$; and the probability of getting from w_2 to w_∞ via μ_b is the probability of utilising the subpath $\mu_b(w_2, w_\infty)$ given that we pass through w_2 , which is $\pi_{\mu_b}(w_\infty \mid w_2)$. As already noted we can express both of these as the products of the probabilities on their component edges, so:

$$\pi_{\mu_a}(w_2 \mid w_0) = \pi_{e_\alpha}(w_1 \mid w_0) \pi_{e_\beta}(w_2 \mid w_1) \quad \text{and}$$

$$\pi_{\mu_b}(w_\infty \mid w_2) = \pi_{e_\gamma}(w_3 \mid w_2) \pi_{e_\delta}(w_\infty \mid w_3)$$

Hence we can see that:

$$\pi(\lambda) = \pi_{\mu_a}(w_2 \mid w_0) \pi_{\mu_b}(w_\infty \mid w_2)$$

This idea can clearly be generalised to $w_0 \rightarrow w_\infty$ paths λ of any length, comprising any number of conjoined subpaths, themselves of any length.

We now proceed to the second proof of Proposition 5.

Alternative Proof of Proposition 5:

Consider a single $w_0 \rightarrow w_\infty$ path λ passing through w_1, w_2 and w_3 . This path can be thought of as four conjoined path-segments $\mu_0(w_0, w_1)$, $\mu_1(w_1, w_2)$, $\mu_2(w_2, w_3)$ and $\mu_3(w_3, w_\infty)$, so:

$$\pi(\lambda) = \pi_{\mu_0}(w_1 \mid w_0) \pi_{\mu_1}(w_2 \mid w_1) \pi_{\mu_2}(w_3 \mid w_2) \pi_{\mu_3}(w_\infty \mid w_3)$$

Considering the event $\Lambda(w_1, w_2, w_3)$, which is the union of all $w_0 \rightarrow w_\infty$ paths passing through the positions w_1, w_2 and w_3 , we get:

$$\begin{aligned} \pi(\Lambda(w_1, w_2, w_3)) &= \sum_{\mu_0, \mu_1, \mu_2, \mu_3} \pi_{\mu_0}(w_1 \mid w_0) \pi_{\mu_1}(w_2 \mid w_1) \pi_{\mu_2}(w_3 \mid w_2) \pi_{\mu_3}(w_\infty \mid w_3) \\ &= \sum_{\mu_0} \pi_{\mu_0}(w_1 \mid w_0) \sum_{\mu_1} \pi_{\mu_1}(w_2 \mid w_1) \sum_{\mu_2} \pi_{\mu_2}(w_3 \mid w_2) \sum_{\mu_3} \pi_{\mu_3}(w_\infty \mid w_3) \\ &= \pi(w_1 \mid w_0) \pi(w_2 \mid w_1) \pi(w_3 \mid w_2) \pi(w_\infty \mid w_3) \end{aligned}$$

using the result from Proposition 4

$$= \pi(\Lambda(w_1)) \times \pi(\Lambda(w_2) \mid \Lambda(w_1)) \times \pi(\Lambda(w_3) \mid \Lambda(w_2)) \times 1$$

Similarly:

$$\pi(\Lambda(w_1, w_2)) = \pi(\Lambda(w_1)) \times \pi(\Lambda(w_2) \mid \Lambda(w_1)) \times 1$$

So:

$$\begin{aligned} \pi(w_3 \mid w_1, w_2) &= \frac{\pi(\Lambda(w_1, w_2), \Lambda(w_3))}{\pi(\Lambda(w_1, w_2))} \\ &= \frac{\pi(\Lambda(w_1, w_2, w_3))}{\pi(\Lambda(w_1, w_2))} \\ &= \frac{\pi(\Lambda(w_1)) \times \pi(\Lambda(w_2) \mid \Lambda(w_1)) \times \pi(\Lambda(w_3) \mid \Lambda(w_2)) \times 1}{\pi(\Lambda(w_1)) \times \pi(\Lambda(w_2) \mid \Lambda(w_1)) \times 1} \\ &= \pi(\Lambda(w_3) \mid \Lambda(w_2)) \\ &= \pi(w_3 \mid w_2) \end{aligned}$$

□

The result of Proposition 5 can readily be extended to give us a class of numerous similar, but slightly different results; for example:

$$\pi(\Lambda(e(w_3, w_4)) \mid \Lambda(w_1), \Lambda(e(w_1, w_2)), \Lambda(w_3)) = \pi(\Lambda(e(w_3, w_4)) \mid \Lambda(w_3))$$

for $w_1 \prec w_2 \prec w_3 \prec w_4$.

This class of results proves very useful when doing causal analysis, and in chapter 5 I will use a number of them without proof. This is simply so that the reader does not get swamped in detail – the proofs of all results in this class follow the proof of Proposition 5 closely, the proof of the above result can be pared down to:

$$\begin{aligned} \pi(\Lambda(e(w_3, w_4)) \mid \Lambda(w_1), \Lambda(e(w_1, w_2)), \Lambda(w_3)) \\ &= \frac{\pi(\Lambda(w_1)) \pi_e(w_2 \mid w_1) \pi(\Lambda(w_3) \mid \Lambda(w_2)) \pi_e(w_4 \mid w_3) \times 1}{\pi(\Lambda(w_1)) \pi_e(w_2 \mid w_1) \pi(\Lambda(w_3) \mid \Lambda(w_2)) \times 1} \\ &= \pi_e(w_4 \mid w_3) \\ &= \pi(\Lambda(e(w_3, w_4)) \mid \Lambda(w_3)) \end{aligned}$$

Corollary 1:

For $w_1 \prec w_2 \prec w_3$:

$$\pi(w_2, w_3 \mid w_1) = \pi(w_3 \mid w_2) \pi(w_2 \mid w_1)$$

Proof:

$$\begin{aligned} \pi(w_2, w_3 \mid w_1) &= \pi(\Lambda(w_2, w_3) \mid \Lambda(w_1)) \\ &= \frac{\pi(\Lambda(w_1), \Lambda(w_2, w_3))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(\Lambda(w_1, w_2, w_3))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(\Lambda(w_1, w_2, w_3))}{\pi(\Lambda(w_1, w_2))} \frac{\pi(\Lambda(w_1, w_2))}{\pi(\Lambda(w_1))} \\ &= \pi(\Lambda(w_3) \mid \Lambda(w_1, w_2)) \pi(\Lambda(w_2) \mid \Lambda(w_1)) \\ &= \pi(w_3 \mid w_1, w_2) \pi(w_2 \mid w_1) \end{aligned}$$

and the result follows from Proposition 5.

□

We are now in a position to prove Results 1 and 2 from section 3.3.

Proposition 6:

For $\Lambda(w_k)$, $Y(w_k)$, $Z(w_k)$ defined as in Definitions 13, 15 and 16:

$$Y(w_k) \amalg Z(w_k) \mid \Lambda(w_k)$$

Proof:

Consider a single element Λ_y in the outcome space of $Y(w_k)$, and the event $\Lambda_y \cap \Lambda(w_k)$. If the criterion variable for w_k is D say, then this event is the union of all $w_0 \rightarrow w_\infty$ paths passing through w_k which utilise edges labelled d_j, e_k, \dots, n_z for some specified j, k, \dots, z (with d_ϕ taking its usual meaning). This can be considered as the union over all $w_0 \rightarrow w_k$ subpaths, of paths each of which consists of a $w_0 \rightarrow w_k$ subpath conjoined to a specific $w_k \rightarrow w_\infty$ subpath.

Consider also a single element Λ_z in the outcome space of $Z(w_k)$ (for which $\Lambda_z \cap \Lambda(w_k) \neq \phi$), and the event $\Lambda_z \cap \Lambda(w_k)$. This event is the union of all $w_0 \rightarrow w_\infty$ paths passing through w_k which utilise edges labelled a_m, b_n, \dots, c_q for some specified m, n, \dots, q (with a_ϕ taking its usual meaning). This can be considered as the union over all $w_k \rightarrow w_\infty$ subpaths, of paths each of which consists of a specified $w_0 \rightarrow w_k$ subpath conjoined to a $w_k \rightarrow w_\infty$ subpath.

Clearly the event $\Lambda_y \cap \Lambda_z \cap \Lambda(w_k)$ will be a single specified $w_0 \rightarrow w_k \rightarrow w_\infty$ path.

Denote this path λ and let its component conjoined path-segments be $\mu_0(w_0, w_k)$ and $\mu_1(w_k, w_\infty)$. Then:

$$\begin{aligned}\pi(\Lambda_y, \Lambda_z, \Lambda(w_k)) &= \pi(\lambda) = \pi_{\mu_0}(w_k \mid w_0) \pi_{\mu_1}(w_\infty \mid w_k) \\ \pi(\Lambda_y, \Lambda(w_k)) &= \sum_{\mu_0} \pi_{\mu_0}(w_k \mid w_0) \pi_{\mu_1}(w_\infty \mid w_k) \\ \pi(\Lambda_z, \Lambda(w_k)) &= \pi_{\mu_0}(w_k \mid w_0) \sum_{\mu_1} \pi_{\mu_1}(w_\infty \mid w_k) = \pi_{\mu_0}(w_k \mid w_0)\end{aligned}$$

since $\sum_{\mu_1} \pi_{\mu_1}(w_\infty \mid w_k) = 1$.

Now $\pi(\Lambda(w_k)) = \pi(w_k \mid w_0) = \sum_{\mu_0} \pi_{\mu_0}(w_k \mid w_0)$ from Proposition 4, so consider:

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda_z, \Lambda(w_k)) &= \frac{\pi(\Lambda_y, \Lambda_z, \Lambda(w_k))}{\pi(\Lambda_z, \Lambda(w_k))} \\ &= \frac{\pi_{\mu_0}(w_k \mid w_0) \pi_{\mu_1}(w_\infty \mid w_k)}{\pi_{\mu_0}(w_k \mid w_0)} \\ &= \pi_{\mu_1}(w_\infty \mid w_k)\end{aligned}$$

Furthermore

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda(w_k)) &= \frac{\pi(\Lambda_y, \Lambda(w_k))}{\pi(\Lambda(w_k))} \\ &= \frac{\sum_{\mu_0} \pi_{\mu_0}(w_k \mid w_0) \pi_{\mu_1}(w_\infty \mid w_k)}{\sum_{\mu_0} \pi_{\mu_0}(w_k \mid w_0)} \\ &= \pi_{\mu_1}(w_\infty \mid w_k)\end{aligned}$$

Hence

$$\pi(\Lambda_y \mid \Lambda_z, \Lambda(w_k)) = \pi(\Lambda_y \mid \Lambda(w_k)) \quad \forall \Lambda_y, \Lambda_z \text{ in the outcome spaces of } Y(w_k), Z(w_k)$$

and

$$Y(w_k) \amalg Z(w_k) \mid \Lambda(w_k)$$

□

Proposition 7:

For $\Lambda(u_m)$, $X(u_m)$, $Z(u_m)$ defined as in Definitions 18, 14 and 17:

$$X(u_m) \amalg Z(u_m) \mid \Lambda(u_m)$$

Proof:

Consider a single element Λ_x in the outcome space of $X(u_m)$, and the event $\Lambda_x \cap \Lambda(u_m)$. If the criterion variable for $\{w\} \in u_m$ is D say, then this event is the union of all $w_0 \rightarrow w_\infty$ paths passing through some $w \in u_m$ which utilise an edge labelled d_j for some specified j . If we let the edge leaving w labelled d_j terminate in the position w_j , then $\Lambda_x \cap \Lambda(u_m)$ can be considered as the union over all $w \in u_m$, over all $w_0 \rightarrow w$ subpaths and over all $w_j \rightarrow w_\infty$ subpaths, of paths each of which consists of a $w_0 \rightarrow w$ subpath conjoined to an edge $e(w, w_j)$ labelled d_j , conjoined to a $w_j \rightarrow w_\infty$ subpath.

Consider also a single element Λ_z in the outcome space of $Z(u_m)$ (for which $\Lambda_z \cap \Lambda(u_m) \neq \phi$), and the event $\Lambda_z \cap \Lambda(u_m)$. This event is the union of all $w_0 \rightarrow w_\infty$ paths passing through some specific $w \in u_m$ which utilise edges labelled a_m, b_n, \dots, c_q for some specified m, n, \dots, q , with b_ϕ taking its usual meaning (NB: If Λ_z is a single element in the outcome space, then each of m, n, \dots, q is specified, and hence one can only pass through a single $w \in u_m$). This event $(\Lambda_z \cap \Lambda(u_m))$ can be considered as the union over all $w \rightarrow w_\infty$ subpaths (for some specific $w \in u_m$), of paths each of which consists of a specified $w_0 \rightarrow w$ subpath conjoined to a $w \rightarrow w_\infty$ subpath.

The event $\Lambda_x \cap \Lambda_z \cap \Lambda(u_m)$ will be the union over all $w_j \rightarrow w_\infty$ subpaths, of paths which consist of a specified $w_0 \rightarrow w$ subpath (for some w) conjoined to an edge $e(w, w_j)$ labelled d_j , conjoined to a $w_j \rightarrow w_\infty$ subpath.

If we denote the specified $w_0 \rightarrow w$ subpath by $\mu_0^*(w_0, w)$ then we have:

$$\begin{aligned} \pi(\Lambda_x, \Lambda_z, \Lambda(u_m)) &= \pi_{\mu_0^*}(w \mid w_0) \pi_e(w_j \mid w) \sum_{\mu} \pi_{\mu}(w_\infty \mid w_j) \\ &= \pi_{\mu_0^*}(w \mid w_0) \pi_e(w_j \mid w) \end{aligned}$$

since $\sum_{\mu} \pi_{\mu}(w_\infty \mid w_j) = 1$

Similarly:

$$\begin{aligned}\pi(\Lambda_x, \Lambda(u_m)) &= \sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \pi_e(w_j \mid w) \sum_{\mu} \pi_{\mu}(w_{\infty} \mid w_j) \right] \\ &= \sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \pi_e(w_j \mid w) \right]\end{aligned}$$

And:

$$\begin{aligned}\pi(\Lambda_z, \Lambda(u_m)) &= \pi_{\mu_0^*}(w \mid w_0) \sum_{\mu'} \pi_{\mu'}(w_{\infty} \mid w) \\ &= \pi_{\mu_0^*}(w \mid w_0)\end{aligned}$$

Also:

$$\pi(\Lambda(u_m)) = \sum_{w \in u_m} \pi(\Lambda(w))$$

from Definition 18

$$= \sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \right]$$

So:

$$\begin{aligned}\pi(\Lambda_x \mid \Lambda_z, \Lambda(u_m)) &= \frac{\pi(\Lambda_x, \Lambda_z, \Lambda(u_m))}{\pi(\Lambda_z, \Lambda(u_m))} \\ &= \frac{\pi_{\mu_0^*}(w \mid w_0) \pi_e(w_j \mid w)}{\pi_{\mu_0^*}(w \mid w_0)} \\ &= \pi_e(w_j \mid w)\end{aligned}$$

And:

$$\begin{aligned}\pi(\Lambda_x \mid \Lambda(u_m)) &= \frac{\pi(\Lambda_x, \Lambda(u_m))}{\pi(\Lambda(u_m))} \\ &= \frac{\sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \pi_e(w_j \mid w) \right]}{\sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \right]}\end{aligned} \tag{3.6.3}$$

But for all $w \in u_m$, $\pi(d_j \mid \Lambda(w)) = \pi(d_j \mid \Lambda(u_m))$ by Proposition 3, and $e(w, w_j)$ is the edge leaving w labelled d_j , so $\pi_e(w_j \mid w) = \pi(d_j \mid \Lambda(w)) = \pi(d_j \mid \Lambda(u_m))$, and so the numerator of expression (3.6.3) is equal to:

$$\left[\sum_{w \in u_m} \left[\sum_{\mu_0} \pi_{\mu_0}(w \mid w_0) \right] \right] \pi(d_j \mid \Lambda(u_m))$$

and so:

$$\begin{aligned}\pi(\Lambda_x \mid \Lambda(u_m)) &= \pi(d_j \mid \Lambda(u_m)) \\ &= \pi_e(w_j \mid w)\end{aligned}$$

Hence:

$$\pi(\Lambda_x \mid \Lambda_z, \Lambda(u_m)) = \pi(\Lambda_x \mid \Lambda(u_m)) \quad \forall \Lambda_x, \Lambda_z \text{ in the outcome spaces of } X(u_m), Z(u_m)$$

and:

$$X(u_m) \amalg Z(u_m) \mid \Lambda(u_m)$$

□

We now have a set of three extremely useful properties:

$$Y(w_k) \amalg Z(w_k) \mid \Lambda(w_k)$$

$$X(u_m) \amalg Z(u_m) \mid \Lambda(u_m)$$

and $\pi(w_3 \mid w_1, w_2) = \pi(w_3 \mid w_2)$ which could be written as:

$$\Lambda(w_3) \amalg \Lambda(w_1) \mid \Lambda(w_2)$$

for $w_1 \prec w_2 \prec w_3$.

In the proofs of Results 1 and 2 we have noted that $\pi(\Lambda(w)) = \sum_{\mu_0} \pi_{\mu_0}(w \mid w_0)$, where the sum is taken over all subpaths $\mu_0(w_0, w)$; and that $\pi(\Lambda(u)) = \sum_{w \in u} \Lambda(w)$: ie. both $\pi(\Lambda(w))$ and $\pi(\Lambda(u))$ are functions of information contained solely within the subpaths $w_0 \rightarrow w$ or $w_0 \rightarrow w \in u$. This is actually rather interesting as $\Lambda(w)$ is defined as the union of all $w_0 \rightarrow w_\infty$ paths passing through w , which would appear to be a function of information contained in the subpaths both upstream and downstream of w .

What we are seeing is that, despite the way they are defined, $\Lambda(w)$ (and $\Lambda(u)$) convey to us no information about the CEG beyond w (or u), and that, as suggested in section 3.3, we can think of $\Lambda(w)$ as the *history* of the position w .

But although $\Lambda(w)$ carries no information about the CEG downstream of w , Result 2 nonetheless tells us that we need nothing except $\Lambda(w)$ (and the distribution of $Y(w) \mid \Lambda(w)$) in order to calculate the probability of all $w \rightarrow w_\infty$ subpaths.

Results 1 and 2 give us almost conventional conditional independence statements — the regular $X \amalg Y \mid Z$ statement where X, Y, Z are (groups of) variables has been replaced by a context-specific statement of the form $X \amalg Y \mid \Lambda$. Recall that what they tell us is that:

- If we know at what stage (u) we are in a process, then the distribution of the possible immediately following steps ($\pi_{X(u)}$) is determined solely by the history of the process upto that stage ($\Lambda(u)$), or alternatively, the information stored in that stage.

- If we know at what position (w) we are in a process, then the distribution of the whole possible further development of the process ($\pi_{Y(w)}$) is determined solely by the history of the process upto that position ($\Lambda(w)$), or alternatively, the information stored in that position.

Our third result is much less conventional, as it gives us a conditional independence statement expressed solely in terms of events. What it tells us is that reaching a position (w_3 in the given expression) given that we have passed through an earlier position (w_2) is independent of the path taken to that earlier position.

In Figure 4 for example, a patient requiring treatment (w_8), given that they are Rhesus +ve (w_4) is independent of whether or not they have blood group B (w_1) or A or AB (w_2) ($\pi(t \mid ym') = \pi(t \mid zm')$). Our exemplar CEGs in this chapter are a little small to allow us to appreciate the power of this result, but its usefulness will become apparent when looking at probability propagation on, and causal analysis of CEGs, in chapters 4 and 5.

We can use these results to express context-specific conditional independencies in asymmetric models, which we could not derive from a BN, even using the d-separation theorem which will allow expression of all (variable-based) independencies if a model is symmetric. As we can use a CEG to represent types of asymmetric model not readily describable via a context-specific BN, we can also express context-specific independencies which would not be yielded by a context-specific BN. However, these results, and those in [53][45] are not yet a d-separation theorem for CEGs (a necessarily more sophisticated animal than that for BNs), so we still have work to do on the reading of CEGs.

4. Probability propagation on Chain Event Graphs

4.1 Introduction

There is a considerable literature on the subject of propagation on Bayesian Networks (propagation of evidence, information, probabilities etc.), as well as on the related topic of Learning BNs. Typically, such propagation (as described in [56] for example) relies on the fact that the analyst can use the conditional independence properties of a BN to reinterpret the joint distribution as a collection of *local* relationships between groups of variables.

The BN tends to undergo some form of modification before any propagation algorithm is applied, and this usually involves the DAG of the BN being *moralised* (by adding undirected edges between the parents of *unmarried* vertices); having its arrows removed; and *triangularised* by adding further edges. Ideally the analyst will be able to represent the resultant graph as a Junction Tree, and it will be to this tree that the propagation algorithm is applied. The algorithm will typically consist of a series of local computations associated with adjacent vertices (corresponding to adjacent cliques in the triangularised BN).

Such algorithms are described in [50] (for the propagation of belief functions), and in [33], wherein it is emphasised that it is unnecessary to fully calculate conditional distributions associated with vertices, and that one can instead work with non-normalised probabilities or *evidence potentials* (simply *potentials* in [56][36] etc.), only calculating normalising constants at the end of the process. This type of algorithm, where one can calculate marginals without computing the joint distribution is usually described as Local Message Passing.

Developments of the basic process include the *penniless propagation* of [5], an adaptation of Shafer-Shenoy propagation in which Probability Trees are used to represent both the message and the potentials stored in the vertices of the Junction Tree. *Lazy propagation*, described in [35] (and also in [6]) attempts to avoid unnecessary computational operations in the propagation algorithm, and allows the updating of probabilities for small subsets of variables given a particular set of evidence.

In [35], Madsen and Jensen suggest that inference algorithms which use the structure of the BN directly (especially the directionality of the edges), are often much more efficient than the Junction Tree algorithms mentioned above. The propagation algorithms described in this chapter work with the CEG topology as it stands without the need for any initial

topological modification as required for Local Message Passing on BNs. *Lazy* shortcuts to our algorithms are described in section 4.3, and as will be seen, the algorithms often simplify the topology of the CEG.

In [36] the authors develop the ideas of Cano, Moral and Salmeron, and note that subtrees of the Probability Trees associated with the vertices of a graph may be *proportional*, and hence that a large Probability Tree can often be represented as the product of two smaller Trees. It should be noted that their *proportionality* is a product of their use of non-normalised posterior probabilities (potentials), and that in fact their proportional subtrees are *identical* in the sense described by me in chapter 2. Martinez, Rodriguez and Salmeron use this idea to simplify calculations in their propagation algorithm. As I have shown earlier in this thesis, the idea can be extended to the creation of a structure (the CEG), whose topology reflects this proportionality.

The process of Learning a BN (as described in for example [18]), involves the production of (and choice between) possible posterior BNs given a prior BN and some data. We are not concerned here with *Learning CEGs*, but note that there are similarities between propagation on CEGs and learning BNs. In our propagation algorithms we assume we know the structure of the CEG, unlike when learning a BN, but other aspects of the process are similar: In learning BNs, *Greedy hillclimbing* [16] involves starting with some initial network and making *local* changes (adding or removing edges) which maximise the improvement in some score. In propagation upon CEGs, our observation does (like in learning BNs) lead to us revising the structure or topology of the CEG, and this revision is essentially a combination of *local* changes such as removing edges or combining positions, similar to those described above.

As mentioned in section 1.2, BNs are not the only graphical model which have been used for propagation. Jaeger's Probabilistic Decision Graph [23][24] was designed for the purpose of fast propagation of information. As already noted this graph cannot express all the conditional independencies of a BN, whereas a CEG can not only do this, but can also express many context-specific conditional independencies that BNs struggle to represent.

In this chapter I focus on the propagation of probabilities on CEGs, by which I mean the updating of the topology and of the edge-probabilities of a CEG, following the observation of some event. I start by considering events whose component paths all pass through a specified collection of positions, before demonstrating how to propagate probabilities following the observation of a more general event. The processes described are also expressed

in the form of Local Message Passing algorithms directly analogous to those produced for BNs.

4.2 Observation of a set of positions

The early sections of this chapter are based on [61] and [62]; and are concerned with updating CEG edge-probabilities following a particular type of observation. The notation has been modified to reflect my interest in root-to-sink paths. Also, in this thesis I have the space to present concepts and techniques more formally than in the afore-mentioned papers.

Recall that we use the notation $w_1 \prec w_2$ to mean that there exists a root-to-sink path passing through w_1 and w_2 with w_1 preceding w_2 on this path. In this section the expressions $w_2 \succ w_1$, $w_1 \preceq w_2$, $w_1 \not\prec w_2$ and $w_1 \not\succ w_2$ will take their obvious meanings.

We start by supposing that an event occurs and is observed; that is we know that we have followed one of a subset of root-to-sink paths within the CEG. The simplest sort of events for our purposes are those whose component paths all pass through a specified collection of positions (and where all paths that pass through these positions are elements (atoms) of our event).

We also require at this stage that no paths in our CEG pass through more than one position in this specified collection of positions. This set of positions can therefore be thought of as a subset of a W -cut across the CEG (see section 3.5).

Any event Λ of this type can be expressed as:

$$\Lambda = \bigcup_{w \in W_X} \Lambda(w)$$

for some collection of positions W_X , where $\Lambda(w_a, w_b) = \phi$ for $w_a, w_b \in W_X$, $w_a \neq w_b$.

Events that can be expressed in this way are often the type one would be analysing if one was working with a BN. For example, in Example 6 in section 3.5, if we were to take $\Lambda = \Lambda(w_5) \cup \Lambda(w_6)$, then this is clearly the event $\{B = b_1\}$, which has meaning both in our CEG and the BN-representation of the CEG (section 3.5 Figure 7).

In the sections of this chapter where we are dealing with a collection of positions W_X , the symbols w_a and w_b will always indicate positions in W_X , and w_1, w_2 etc. will always indicate positions not in W_X . Throughout the chapter I will also use the symbol w for brevity, but whenever this symbol is used, I will specify whether or not such positions

have any especial characteristics; so for example in sections 4.2 and 4.3 I will be using w for general positions in the set W_X .

Definition 24: Consider a position w_1 such that $w_1 \not\prec w$ for any $w \in W_X$. Then let:

$$\begin{aligned}\pi(\Lambda \mid w_1) &= \pi\left(\bigcup_{w \in W_X} \Lambda(w) \mid \Lambda(w_1)\right) \\ &= \sum_{w \in W_X} \pi(\Lambda(w) \mid \Lambda(w_1))\end{aligned}$$

since the requirement that no paths in our CEG pass through more than one $w \in W_X$ means that the events $\{\Lambda(w, w_1)\}_{w \in W_X}$ are disjoint.

Now $w_1 \not\prec w$ for any $w \in W_X$ means that for specified w , either:

- $w_1 \prec w$, or
- there is no root-to-sink path in our CEG which passes through w_1 and w ; that is:

$$\begin{aligned}\Lambda(w, w_1) &= \phi \\ \Rightarrow \pi(\Lambda(w) \mid \Lambda(w_1)) &= 0\end{aligned}$$

Hence:

$$\pi(\Lambda \mid w_1) = \sum_{w \in W_X, w \succ w_1} \pi(\Lambda(w) \mid \Lambda(w_1))$$

Let any updated probability in our CEG (following observation of our event), be denoted $\hat{\pi}$. These probabilities are essentially posterior probabilities, so if $w_1 \not\prec w$ for any $w \in W_X$, for example, we can write:

$$\begin{aligned}\hat{\pi}(w_1) &= \pi(w_1 \mid \Lambda) \\ &= \pi(\Lambda(w_1) \mid \Lambda) \\ &= \frac{\pi(\Lambda \mid \Lambda(w_1)) \pi(\Lambda(w_1))}{\pi(\Lambda)} \\ &= \frac{\sum_{w \in W_X} \pi(\Lambda(w) \mid \Lambda(w_1)) \pi(\Lambda(w_1))}{\sum_{w \in W_X} \pi(\Lambda(w))}\end{aligned}$$

since the events $\{\Lambda(w, w_1)\}_{w \in W_X}$ are disjoint.

Proposition 8:

For w_1, w_2 such that $w_1 \prec w_2$ and $w_2 \not\prec w$ for any $w \in W_X$:

$$\hat{\pi}(w_2 \mid w_1) = \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi(w_2 \mid w_1)$$

Proof:

$$\begin{aligned}
\hat{\pi}(w_2 \mid w_1) &= \pi(w_2 \mid \Lambda, w_1) \\
&= \pi(\Lambda(w_2) \mid \Lambda, \Lambda(w_1)) \\
&= \frac{\pi(\Lambda, \Lambda(w_1), \Lambda(w_2))}{\pi(\Lambda, \Lambda(w_1))} \\
&= \frac{\pi(\Lambda \mid \Lambda(w_1), \Lambda(w_2)) \pi(\Lambda(w_1), \Lambda(w_2))}{\pi(\Lambda \mid \Lambda(w_1)) \pi(\Lambda(w_1))} \\
&= \frac{\sum_{w \in W_X} \pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(w_2)) \pi(\Lambda(w_2) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))}
\end{aligned}$$

since the events $\{\Lambda(w, w_1, w_2)\}_{w \in W_X}$ are disjoint.

Now, for given $w \in W_X$, if there is no root-to-sink path passing through w_1, w_2 and w then $\pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(w_2)) = 0$, so the summation in the numerator of this expression can be taken over $w \in W_X$ such that $w_1 \prec w_2 \prec w$.

But, from Proposition 5 in chapter 3, if $w_1 \prec w_2 \prec w$, then:

$$\pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(w_2)) = \pi(\Lambda(w) \mid \Lambda(w_2))$$

Hence:

$$\hat{\pi}(w_2 \mid w_1) = \frac{\sum_{w \in W_X, w_1 \prec w_2 \prec w} \pi(\Lambda(w) \mid \Lambda(w_2)) \pi(\Lambda(w_2) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))}$$

Now recall that w_1 and w_2 are specified positions, and that $w_1 \prec w_2$, so:

$$\{w \in W_X \mid w_1 \prec w_2 \prec w\} \equiv \{w \in W_X \mid w_2 \prec w\}$$

Hence:

$$\begin{aligned}
\hat{\pi}(w_2 \mid w_1) &= \frac{\sum_{w \in W_X, w \succ w_2} \pi(\Lambda(w) \mid \Lambda(w_2)) \pi(\Lambda(w_2) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))} \\
&= \frac{\pi(\Lambda \mid w_2) \pi(w_2 \mid w_1)}{\pi(\Lambda \mid w_1)}
\end{aligned}$$

□

Proposition 9:

For specified $w_b \in W_X$ and $w_1 \not\succ w$ for any $w \in W_X$:

$$\hat{\pi}(w_b \mid w_1) = \frac{\pi(w_b \mid w_1)}{\pi(\Lambda \mid w_1)}$$

Proof:

$$\begin{aligned}
\hat{\pi}(w_b \mid w_1) &= \pi(w_b \mid \Lambda, w_1) \\
&= \pi(\Lambda(w_b) \mid \Lambda, \Lambda(w_1)) \\
&= \frac{\pi(\Lambda, \Lambda(w_1), \Lambda(w_b))}{\pi(\Lambda, \Lambda(w_1))} \\
&= \frac{\pi(\bigcup_{w \in W_X} \Lambda(w), \Lambda(w_1), \Lambda(w_b))}{\pi(\Lambda, \Lambda(w_1))} \\
&= \frac{\pi(\bigcup_{w \in W_X} (\Lambda(w, w_1, w_b)))}{\pi(\Lambda, \Lambda(w_1))}
\end{aligned}$$

Now for $w = w_b$, $\Lambda(w, w_1, w_b) = \Lambda(w_1, w_b)$, and for $w \neq w_b$, $\Lambda(w, w_1, w_b) = \phi$, since we have required that no paths in our CEG pass through more than one $w \in W_X$. So:

$$\begin{aligned}
\hat{\pi}(w_b \mid w_1) &= \frac{\pi(\Lambda(w_1, w_b))}{\pi(\Lambda, \Lambda(w_1))} \\
&= \frac{\pi(\Lambda(w_b) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))} \\
&= \frac{\pi(w_b \mid w_1)}{\pi(\Lambda \mid w_1)} \quad \square
\end{aligned}$$

With my emphasis in this thesis being on root-to-sink paths, it might appear sensible to find an analogue of Proposition 8 for such paths. In fact we will find that such an analogue is not very useful, but it is possible to produce a generalised version of the proposition which works for path-segments.

Corollary 2:

For $w_1 \prec w_2$ such that $w_2 \not\asymp w$ for any $w \in W_X$, and specified path-segment $\mu(w_1, w_2)$:

$$\hat{\pi}_\mu(w_2 \mid w_1) = \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi_\mu(w_2 \mid w_1)$$

Proof:

$$\hat{\pi}_\mu(w_2 \mid w_1) = \hat{\pi}(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))$$

(Result 4 from section 3.6)

$$\begin{aligned}
&= \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda, \Lambda(w_1)) \\
&= \frac{\pi(\Lambda, \Lambda(w_1), \Lambda(\mu(w_1, w_2)))}{\pi(\Lambda, \Lambda(w_1))} \\
&= \frac{\pi(\Lambda \mid \Lambda(w_1), \Lambda(\mu(w_1, w_2))) \pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)))}{\pi(\Lambda \mid \Lambda(w_1)) \pi(\Lambda(w_1))} \\
&= \frac{\sum_{w \in W_X} \pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(\mu(w_1, w_2))) \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))}
\end{aligned}$$

since the events $\{\Lambda(w, w_1, \mu(w_1, w_2))\}$ are disjoint.

Now, as in the proof of Proposition 8, $\pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(\mu(w_1, w_2))) = 0$ if there is no root-to-sink path passing through w_1, w_2 and w . So consider a position w such that $w_1 \prec w_2 \prec w$, and:

$$\begin{aligned}\pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(\mu(w_1, w_2))) &= \frac{\pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)), \Lambda(w))}{\pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2)))} \\ &= \frac{\pi(\Lambda(w_1, \mu(w_1, w_2), w))}{\pi(\Lambda(w_1, \mu(w_1, w_2)))}\end{aligned}$$

NB: That $\pi(\Lambda(w_1), \Lambda(\mu(w_1, w_2))) = \pi(\Lambda(w_1, \mu(w_1, w_2)))$ etc. can be shown in much the same way as that $\Lambda(w_x, w_y) = \Lambda(w_x) \cap \Lambda(w_y)$.

As in the first proof of Proposition 5 in section 3.6, we can think of the event $\Lambda(w_1, \mu(w_1, w_2))$ as the union of all $w_0 \rightarrow w_\infty$ paths passing through w_1 , the specified path-segment $\mu(w_1, w_2)$ and w_2 . Similarly $\Lambda(w_1, \mu(w_1, w_2), w)$ is the union of all $w_0 \rightarrow w_\infty$ paths passing through w_1 , the specified path-segment $\mu(w_1, w_2)$, w_2 and w . So, following the afore-mentioned proof (but omitting much of the detail for brevity), we get:

$$\begin{aligned}\pi(\Lambda(w_1, \mu(w_1, w_2))) &= \pi(w_1 \mid w_0) \pi_\mu(w_2 \mid w_1) \pi(w_\infty \mid w_2) \\ &= \pi(\Lambda(w_1)) \times \pi_\mu(w_2 \mid w_1) \times 1\end{aligned}$$

And:

$$\begin{aligned}\pi(\Lambda(w_1, \mu(w_1, w_2), w)) &= \pi(w_1 \mid w_0) \pi_\mu(w_2 \mid w_1) \pi(w \mid w_2) \pi(w_\infty \mid w) \\ &= \pi(\Lambda(w_1)) \times \pi_\mu(w_2 \mid w_1) \times \pi(\Lambda(w) \mid \Lambda(w_2)) \times 1\end{aligned}$$

So:

$$\begin{aligned}\pi(\Lambda(w) \mid \Lambda(w_1), \Lambda(\mu(w_1, w_2))) &= \frac{\pi(\Lambda(w_1)) \times \pi_\mu(w_2 \mid w_1) \times \pi(\Lambda(w) \mid \Lambda(w_2)) \times 1}{\pi(\Lambda(w_1)) \times \pi_\mu(w_2 \mid w_1) \times 1} \\ &= \pi(\Lambda(w) \mid \Lambda(w_2))\end{aligned}$$

This result should not surprise us, as I stated in chapter 3 that reaching a position (w say) given that we have passed through an earlier position (w_2 say) is independent of the path taken to that earlier position.

So we have:

$$\hat{\pi}_\mu(w_2 \mid w_1) = \frac{\sum_{w \in W_X, w_1 \prec w_2 \prec w} \pi(\Lambda(w) \mid \Lambda(w_2)) \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))}$$

which using the same argument as in the proof of Proposition 8, is:

$$\begin{aligned}&= \frac{\sum_{w \in W_X, w \succ w_2} \pi(\Lambda(w) \mid \Lambda(w_2)) \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))} \\ &= \frac{\pi(\Lambda \mid \Lambda(w_2)) \pi(\Lambda(\mu(w_1, w_2)) \mid \Lambda(w_1))}{\pi(\Lambda \mid \Lambda(w_1))} \\ &= \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi_\mu(w_2 \mid w_1)\end{aligned}$$

□

Corollary 3:

For $w_1 \sim w_2$, $w_1 \prec w_2$, $w_2 \not\asymp w$ for any $w \in W_X$, and specified edge $e(w_1, w_2)$:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi_e(w_2 \mid w_1) \quad (1)$$

$$= \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \hat{\pi}(w_2 \mid w_1) \quad (2)$$

Proof:

Result (1) follows immediately from Corollary 2 by considering $e(w_1, w_2)$ as a path-segment of length 1 edge. Also:

$$\begin{aligned} \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi_e(w_2 \mid w_1) &= \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi(w_2 \mid w_1) \times \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \\ &= \hat{\pi}(w_2 \mid w_1) \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \quad \square \end{aligned}$$

We are now in a position to update the probabilities of the paths within the CEG, but it transpires that for practical purposes it is actually more sensible to update edges one at a time using the results of Propositions 8 and 9 for pairs of adjacent positions, and Corollary 3 when there is more than one edge joining any two adjacent positions.

I briefly describe two further useful results before specifying the algorithm:

We have considered positions w_1 such that $w_1 \prec w$ for some $w \in W_X$. We need also to consider positions w_1 such that:

- $w_1 \not\prec w$ and $w_1 \not\asymp w$ for any $w \in W_X$
- $w_1 \succ w_a$ for some $w_a \in W_X$

Lemma 1:

For $w_1 \not\prec w$, $w_1 \not\asymp w$ for any $w \in W_X$:

$$\hat{\pi}(\Lambda(w_1)) = 0$$

Proof:

$$\begin{aligned} \hat{\pi}(\Lambda(w_1)) &= \pi(\Lambda(w_1) \mid \Lambda) = \frac{\pi(\Lambda, \Lambda(w_1))}{\pi(\Lambda)} \\ &= \frac{\sum_{w \in W_X} \pi(\Lambda(w), \Lambda(w_1))}{\pi(\Lambda)} \end{aligned}$$

since the events $\{\Lambda(w, w_1)\}$ are disjoint

And if $w_1 \not\prec w$ and $w_1 \not\asymp w$ then $\Lambda(w, w_1) = \Lambda(w) \cap \Lambda(w_1) = \phi$, and this expression equals zero.

□

Proposition 10:

For $w_2 \succ w_a$ for some $w_a \in W_X$:

$$\hat{\pi}_\mu(w_2 \mid w_a) = \pi_\mu(w_2 \mid w_a)$$

Proof:

$$\begin{aligned} \hat{\pi}_\mu(w_2 \mid w_a) &= \hat{\pi}(\Lambda(\mu(w_a, w_2)) \mid \Lambda(w_a)) \\ &= \pi(\Lambda(\mu(w_a, w_2)) \mid \Lambda, \Lambda(w_a)) \\ &= \frac{\pi(\Lambda, \Lambda(w_a), \Lambda(\mu(w_a, w_2)))}{\pi(\Lambda, \Lambda(w_a))} \\ &= \frac{\pi(\bigcup_{w \in W_X} \Lambda(w), \Lambda(w_a), \Lambda(\mu(w_a, w_2)))}{\pi(\bigcup_{w \in W_X} \Lambda(w), \Lambda(w_a))} \\ &= \frac{\pi(\bigcup_{w \in W_X} \Lambda(w, w_a, \mu(w_a, w_2))}{\pi(\bigcup_{w \in W_X} \Lambda(w, w_a))} \end{aligned}$$

But if $w = w_a$ then $\Lambda(w, w_a) = \Lambda(w_a)$, $\Lambda(w, w_a, \mu(w_a, w_2)) = \Lambda(w_a, \mu(w_a, w_2))$; and if $w \neq w_a$ then $\Lambda(w, w_a) = \phi$, $\Lambda(w, w_a, \mu(w_a, w_2)) = \phi$, since we have required that no paths in our CEG pass through more than one $w \in W_X$. So:

$$\begin{aligned} \hat{\pi}_\mu(w_2 \mid w_a) &= \frac{\pi(\Lambda(w_a, \mu(w_a, w_2)))}{\pi(\Lambda(w_a))} \\ &= \pi(\Lambda(\mu(w_a, w_2)) \mid \Lambda(w_a)) \\ &= \pi_\mu(w_2 \mid w_a) \quad \text{true for all } w_2 \succ w_a \end{aligned}$$

□

Corollary 4:

Any edge on a path-segment $\mu(w, w_\infty)$ (with $w \in W_X$) has the same probability post-observation as pre-observation.

Proof:

This follows immediately from Proposition 10.

Algorithm

Following our observation of $\Lambda = \bigcup_{w \in W_X} \Lambda(w)$, we update the edge-probabilities on our CEG as follows:

- (1) Remove all undirected edges from the CEG
- (2) If a position w_1 does not lie on a w_0 to $w \in W_X$ to w_∞ path then w_1 and all edges entering or leaving w_1 are pruned (Lemma 1)
- (3) All edges on path-segments $\mu(w, w_\infty)$ (with $w \in W_X$) retain their probabilities from before (Corollary 4)

We now work back from each $w \in W_X$ towards w_0 , updating each $w_0 \rightarrow w$ path-segment edge by edge:

(4) For $w_1 \notin W_X$, $w \in W_X$, $w_1 \sim w$ (ie. w_1 adjacent to w), $w_1 \prec w$:

$$\hat{\pi}(w \mid w_1) = \frac{\pi(w \mid w_1)}{\pi(\Lambda \mid w_1)}$$

(Proposition 9)

If there is more than one edge joining w_1 to w , then:

$$\hat{\pi}_e(w \mid w_1) = \frac{\pi_e(w \mid w_1)}{\pi(w \mid w_1)} \hat{\pi}(w \mid w_1)$$

(Corollary 3)

(5) For $w_1 \prec w_2 \prec w \in W_X$, $w_1 \sim w_2$ (ie. w_1 adjacent to w_2):

$$\hat{\pi}(w_2 \mid w_1) = \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi(w_2 \mid w_1)$$

(Proposition 8)

If there is more than one edge joining w_1 to w_2 , then:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \hat{\pi}(w_2 \mid w_1)$$

(Corollary 3)

Once we have updated all edge-probabilities on all $w_0 \rightarrow w \in W_X$ path-segments, we need to check that our resultant graph is a CEG:

(6) In our updated graph any positions which are now equivalent are combined into a single position. Any positions which are now stage-equivalent are connected by an undirected edge, and edges are coloured as appropriate (see Definition 13).

4.3. An Example

Example 1: Another look at the Blood condition example

To illustrate the ideas of section 4.2 we return to the Blood Condition example from chapters 2 and 3. Figure 1 gives the original CEG, and Figure 2 shows the CEG with pre-observation probabilities. Note that:

$$\sum_{i=1}^4 \theta_i = \sum_{i=6}^8 \theta_i = \sum_{i=9}^{11} \theta_i = 1$$

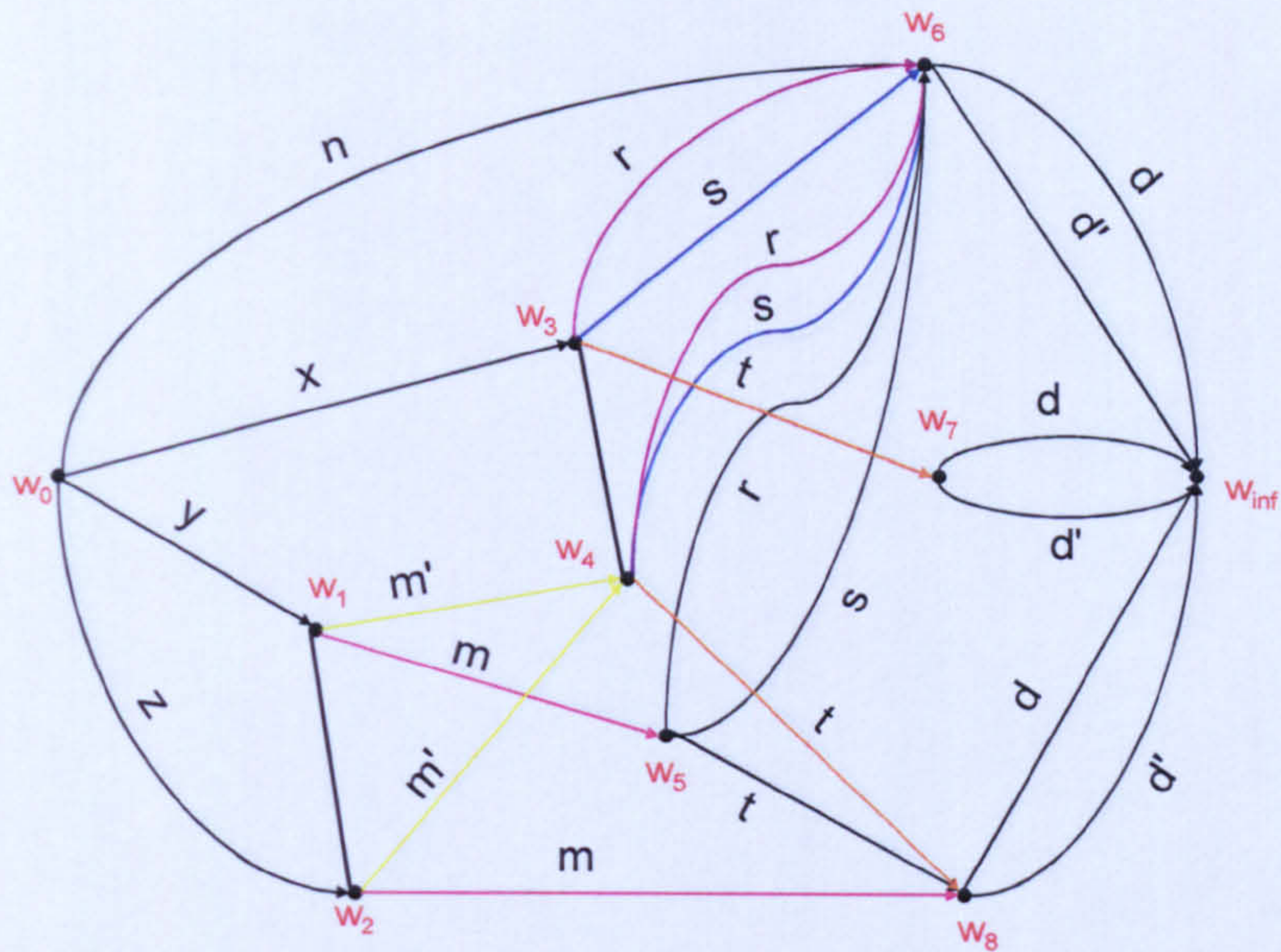


Figure 1: CEG for Blood condition example

We noted in section 3.4 that it was probable that all patients in the group zm would require treatment and that therefore $\Lambda(w_6)$ represented all patients who did not require treatment. If we accept this reading then observing the event $\Lambda = \Lambda(w_6)$ is equivalent to observing that a patient does not require treatment.

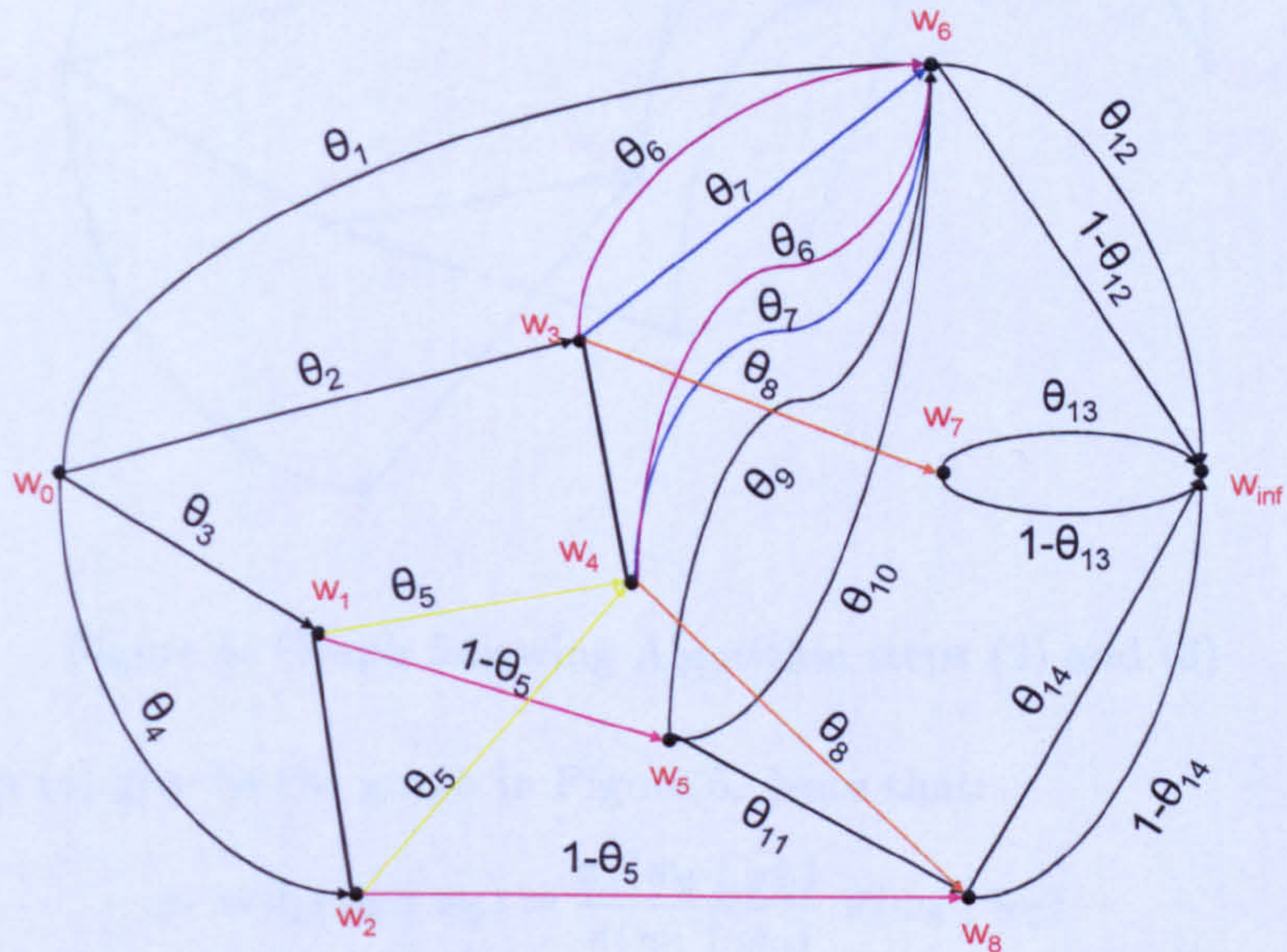


Figure 2: CEG with pre-observation probabilities

Can we use the algorithm from section 4.2 to update the probabilities in our CEG having observed this event?

Algorithm step (1) gives us the graph in Figure 3, and steps (2) and (3) the graph in Figure 4.

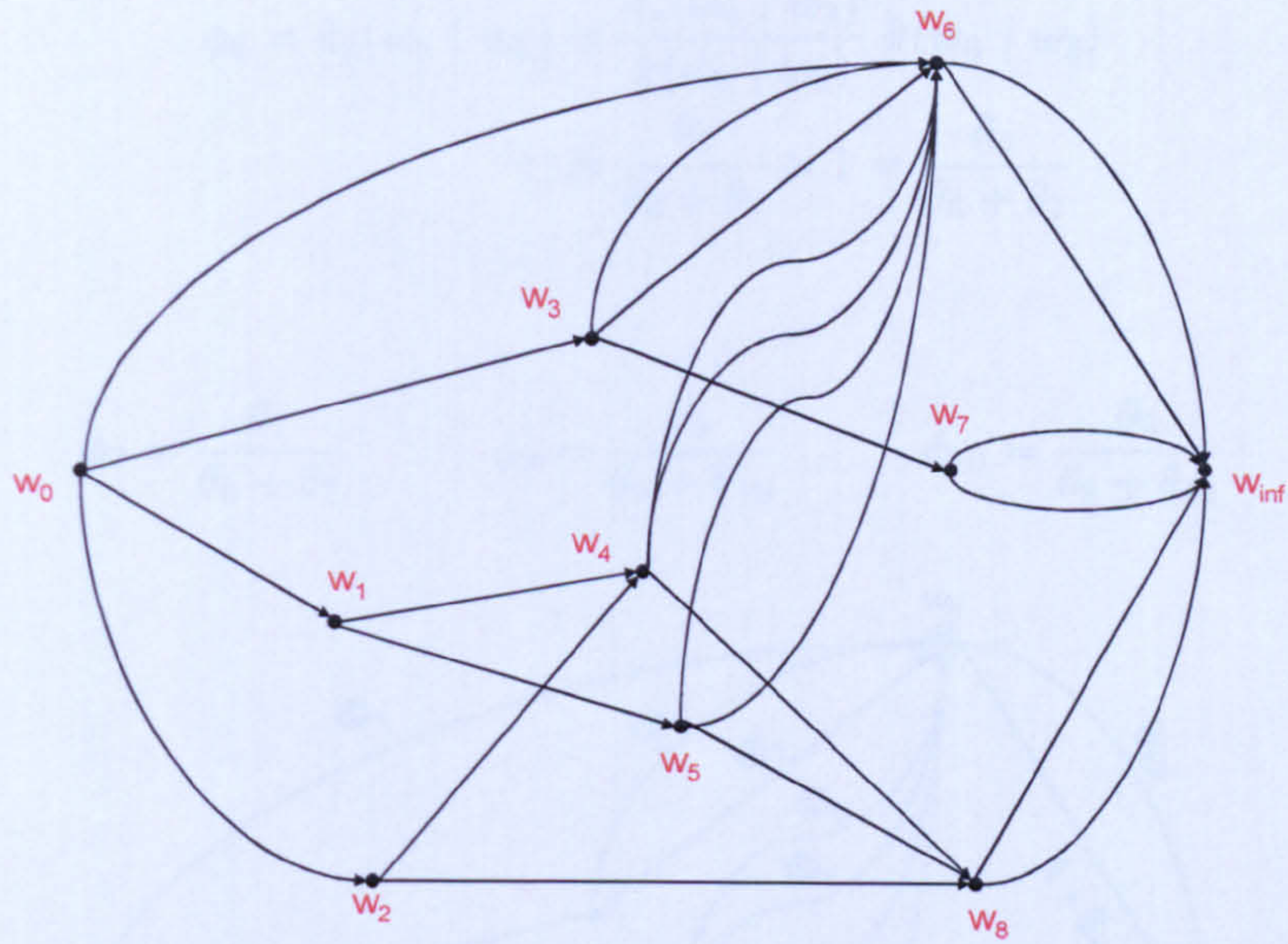


Figure 3: Graph following Algorithm step (1)

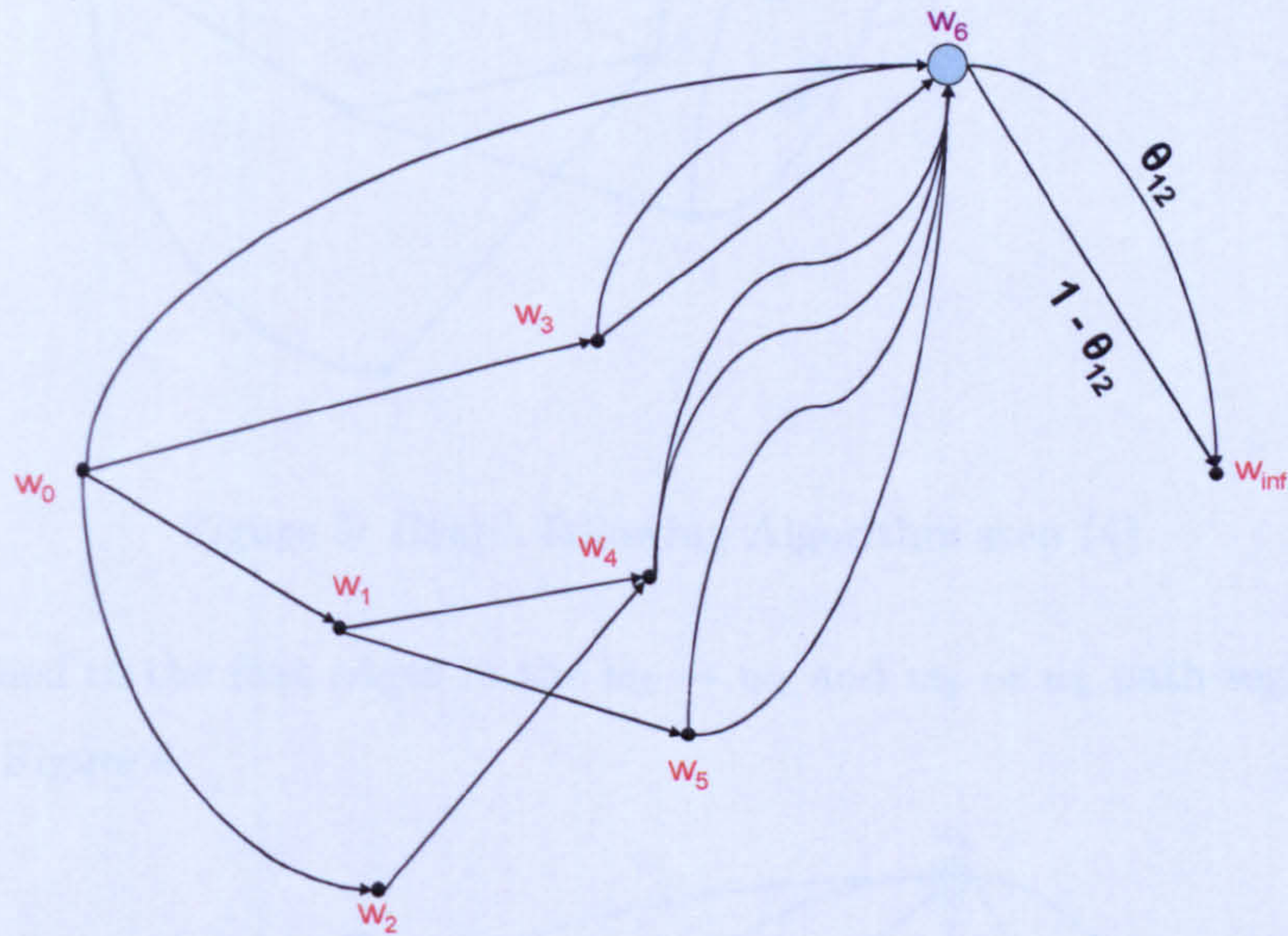


Figure 4: Graph following Algorithm steps (2) and (3)

Algorithm step (4) give us the graph in Figure 5. Note that:

$$\begin{aligned}
 (i) \quad \phi_1 = \hat{\pi}_e(w_6 \mid w_0) &= \frac{\pi_e(w_6 \mid w_0)}{\pi(w_6 \mid w_0)} \hat{\pi}(w_6 \mid w_0) \\
 &= \frac{\theta_1}{\pi(\Lambda(w_6))} \times 1 = \frac{\theta_1}{\pi(\Lambda(w_6))}
 \end{aligned}$$

where $\pi(\Lambda(w_6)) = \theta_1 + \theta_2(\theta_6 + \theta_7) + \theta_3\theta_5(\theta_6 + \theta_7) + \theta_3(1 - \theta_5)(\theta_9 + \theta_{10}) + \theta_4\theta_5(\theta_6 + \theta_7)$.

$$(ii) \quad \hat{\pi}(w_6 \mid w_3) = \frac{\pi(w_6 \mid w_3)}{\pi(\Lambda \mid w_3)} = 1$$

and for the **upper** edge $e(w_3, w_6)$:

$$\begin{aligned}\phi_6 &= \hat{\pi}_e(w_6 \mid w_3) = \frac{\pi_e(w_6 \mid w_3)}{\pi(w_6 \mid w_3)} \hat{\pi}(w_6 \mid w_3) \\ &= \frac{\theta_6}{\theta_6 + \theta_7} \times 1 = \frac{\theta_6}{\theta_6 + \theta_7}\end{aligned}$$

Similarly:

$$\phi_7 = \frac{\theta_7}{\theta_6 + \theta_7} \quad \phi_9 = \frac{\theta_9}{\theta_9 + \theta_{10}} \quad \phi_{10} = \frac{\theta_{10}}{\theta_9 + \theta_{10}}$$

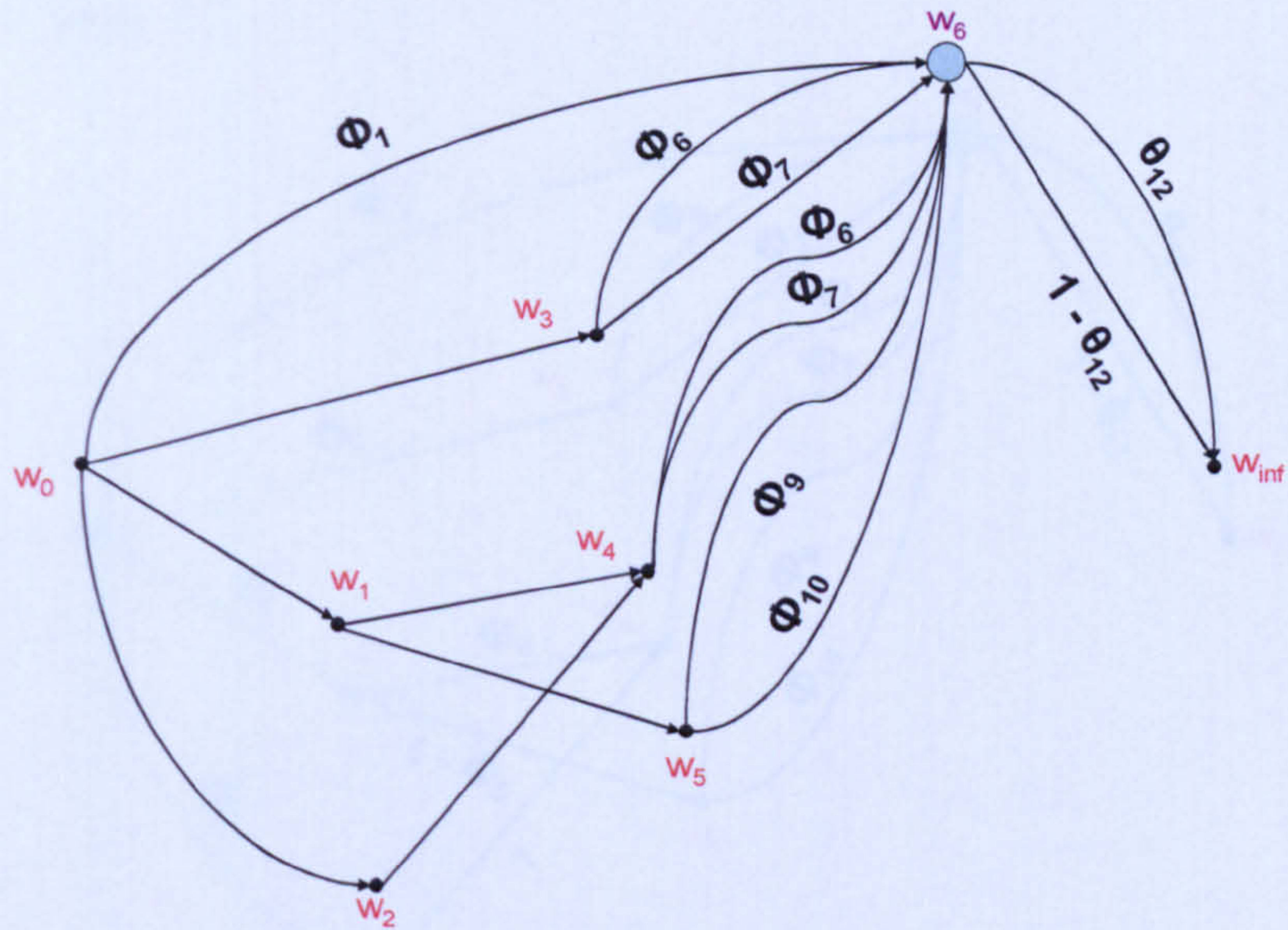


Figure 5: Graph following Algorithm step (4)

Step (5) applied to the first edges in the $w_0 \rightarrow w_4$ and $w_0 \rightarrow w_5$ path-segments gives us the graph in Figure 6:

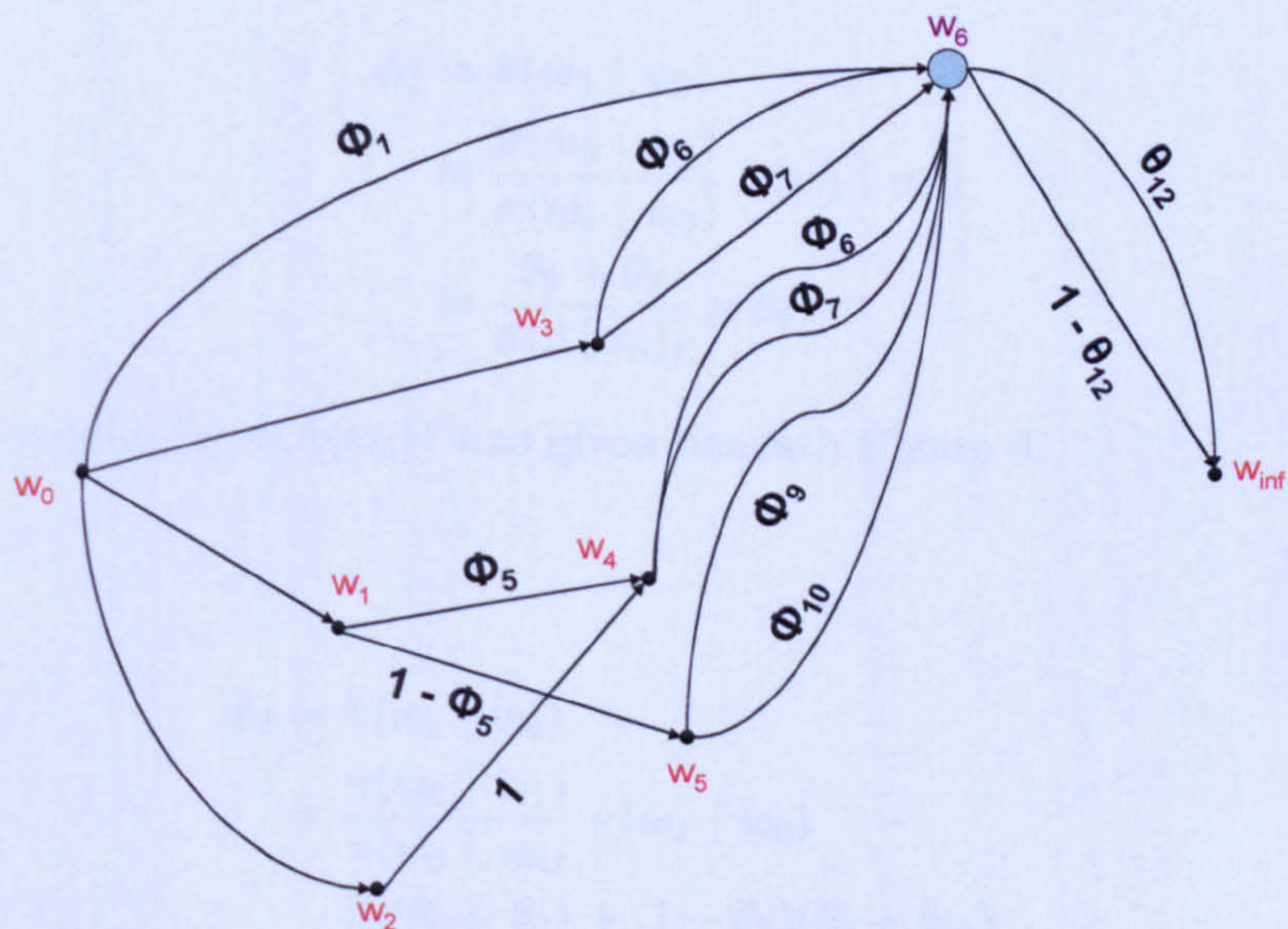


Figure 6: Graph following first application of Algorithm step (5)

Note that:

$$\begin{aligned}
 \phi_5 &= \hat{\pi}(w_4 \mid w_1) \\
 &= \frac{\pi(w_6 \mid w_4)}{\pi(w_6 \mid w_1)} \pi(w_4 \mid w_1) \\
 &= \frac{\theta_6 + \theta_7}{\theta_5(\theta_6 + \theta_7) + (1 - \theta_5)(\theta_9 + \theta_{10})} \times \theta_5
 \end{aligned}$$

Once we have applied step (5) to the final edges in the $w_0 \rightarrow w_6$ path-segments we get the graph in Figure 7:

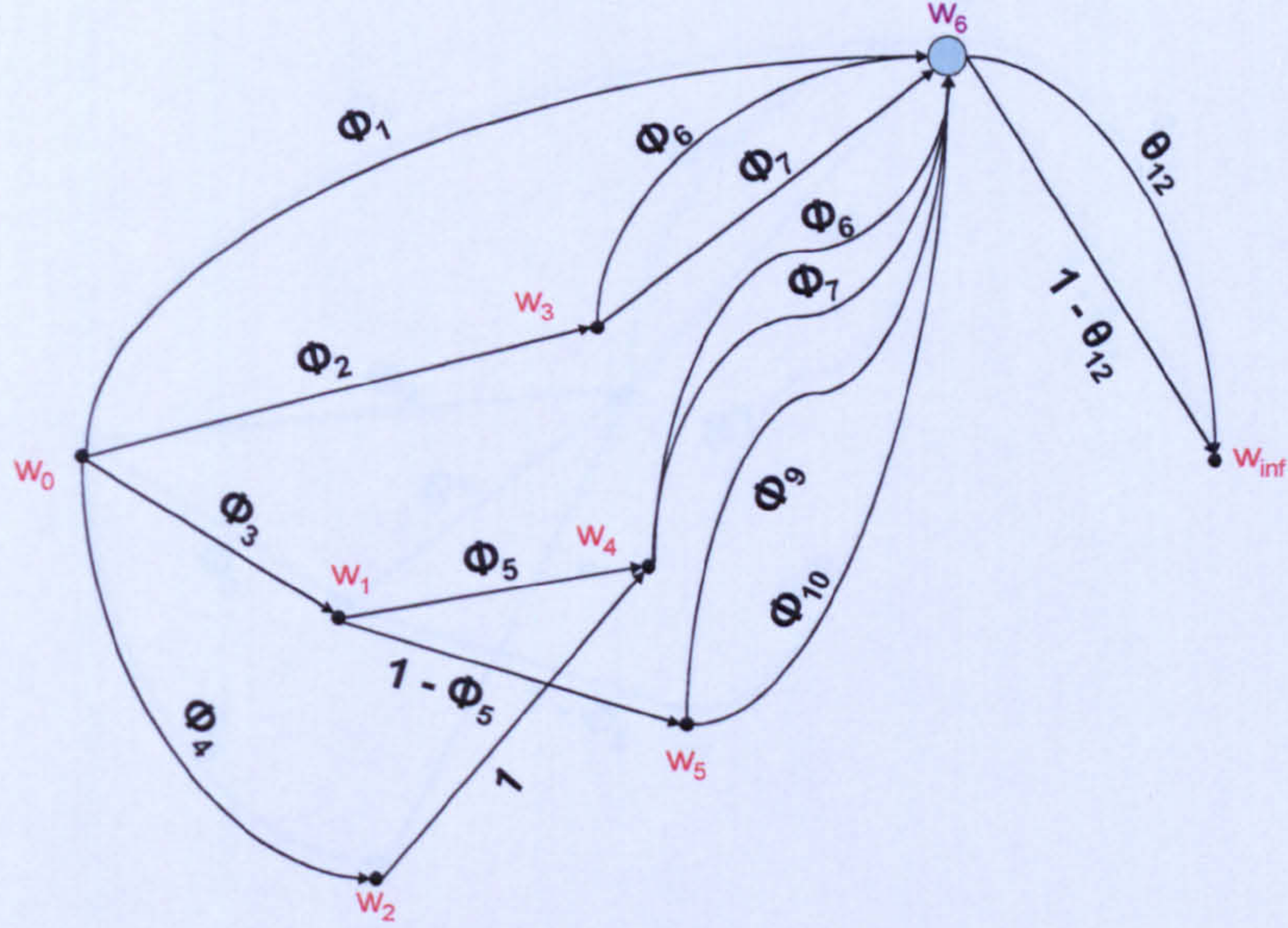


Figure 7: Graph following final application of Algorithm step (5)

Note that:

$$\begin{aligned}
 \phi_2 &= \hat{\pi}(w_3 \mid w_0) \\
 &= \frac{\pi(w_6 \mid w_3)}{\pi(w_6 \mid w_0)} \pi(w_3 \mid w_0) \\
 &= \frac{\theta_6 + \theta_7}{\pi(\Lambda(w_6))} \times \theta_2
 \end{aligned}$$

where an expression for $\pi(\Lambda(w_6))$ was given beneath Figure 4.

Similarly:

$$\begin{aligned}
 \phi_3 &= \hat{\pi}(w_1 \mid w_0) \\
 &= \frac{\pi(w_6 \mid w_1)}{\pi(w_6 \mid w_0)} \pi(w_1 \mid w_0) \\
 &= \frac{\theta_5(\theta_6 + \theta_7) + (1 - \theta_5)(\theta_9 + \theta_{10})}{\pi(\Lambda(w_6))} \times \theta_3 \\
 \phi_4 &= \hat{\pi}(w_2 \mid w_0)
 \end{aligned}$$

$$\begin{aligned}
&= \frac{\pi(w_6 \mid w_2)}{\pi(w_6 \mid w_0)} \pi(w_2 \mid w_0) \\
&= \frac{\theta_5(\theta_6 + \theta_7)}{\pi(\Lambda(w_6))} \times \theta_4
\end{aligned}$$

We note that the positions w_1 and w_2 are no longer stage-equivalent, but that the positions w_3 and w_4 which previously were stage-equivalent are now equivalent, so step (6) gives us the CEG in Figure 8. Note that the combining of positions w_3 and w_4 requires us to relabel the positions w_5 and w_6 .

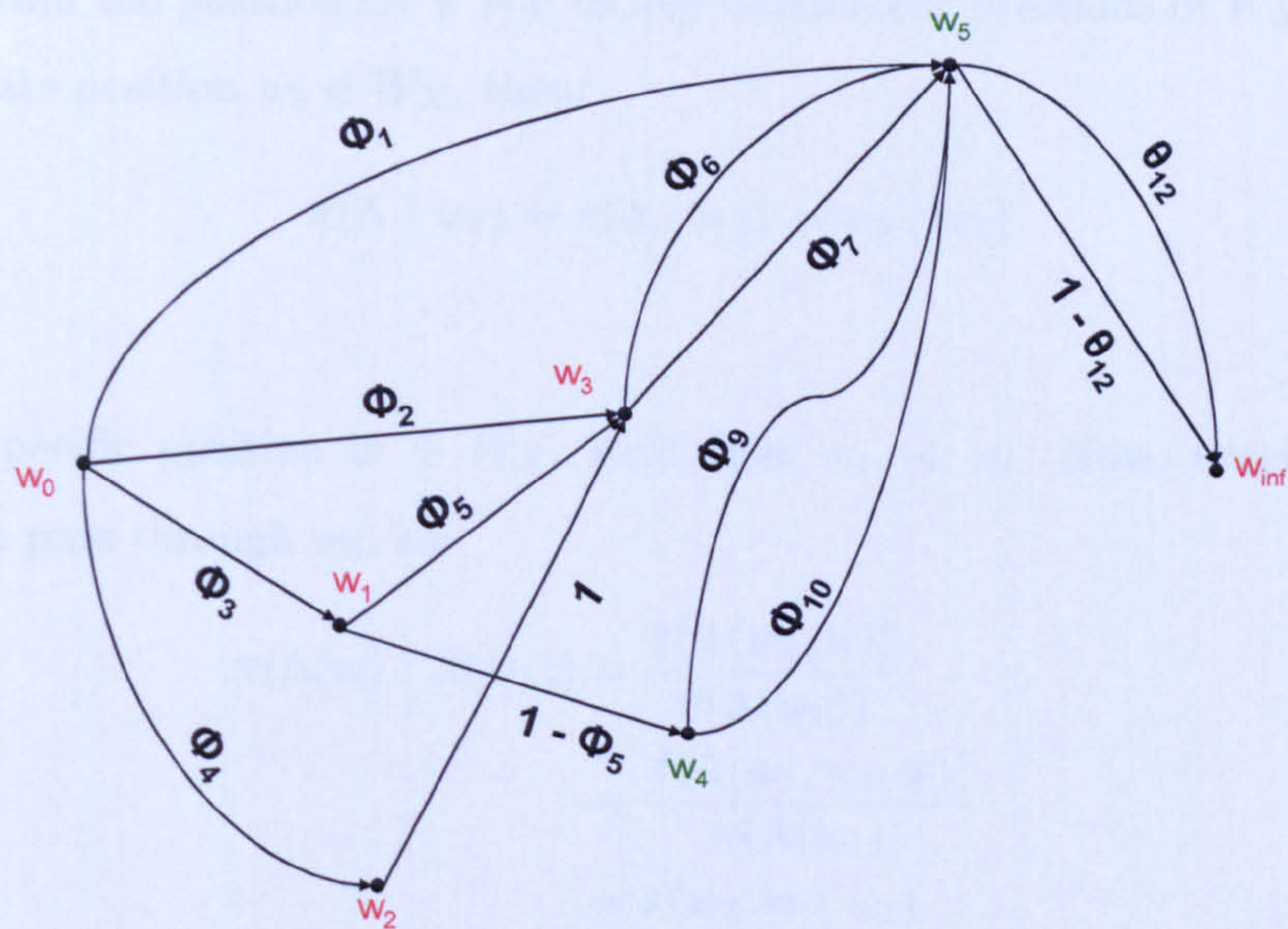


Figure 8: CEG following completion of our algorithm

Figure 9 shows our updated CEG in the same format as the original CEG in Figure 2.

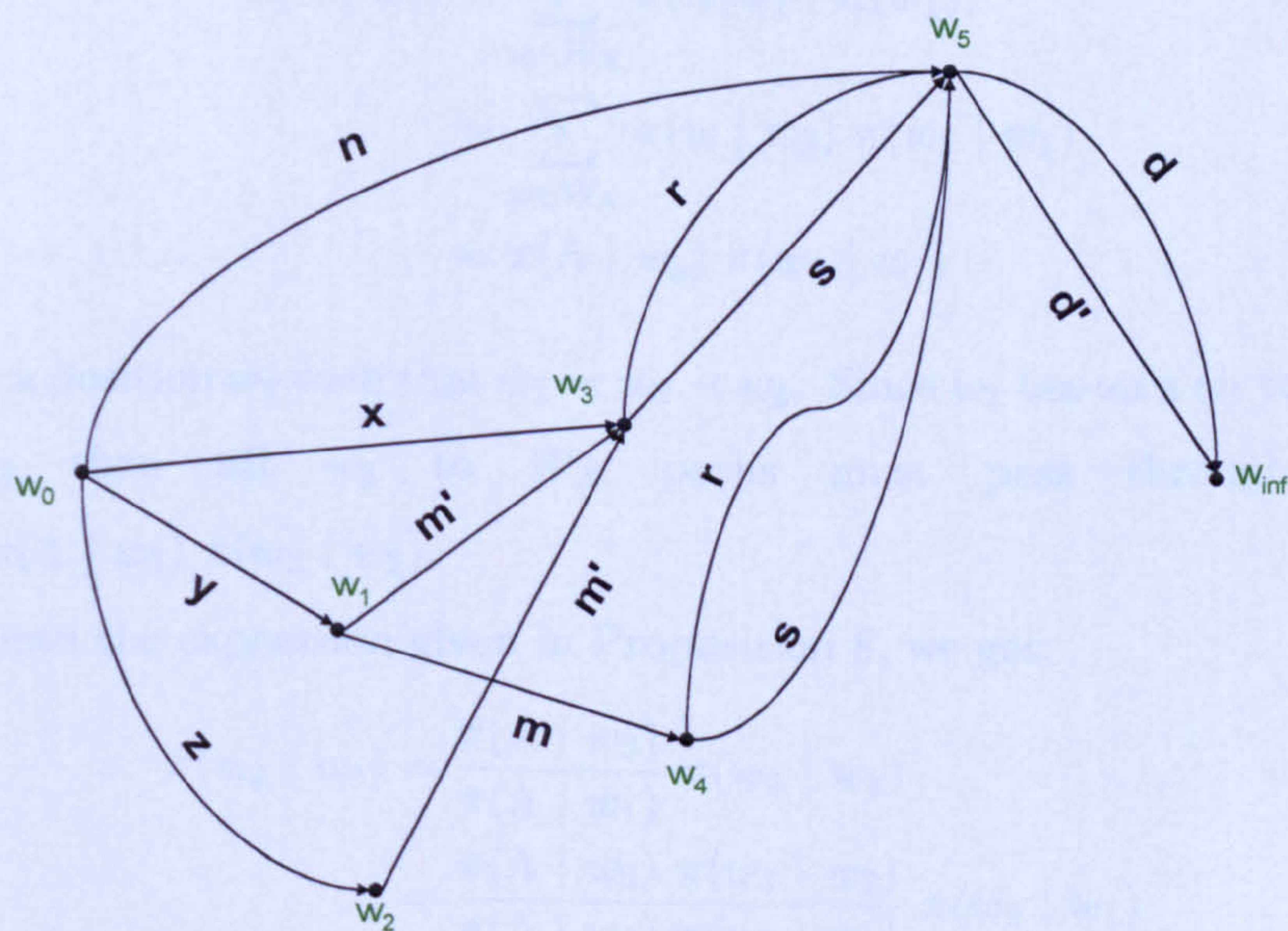


Figure 9: Final CEG

4.4 Lazy short-cuts

There are several highly useful refinements that we can make to the algorithm given in section 4.2 which allow us to avoid unnecessary computational operations. In this sense they are analogous to the refinements present in Lazy propagation on BNs ([35][6]). These refinements rely on the result given in the following corollary:

Corollary 5:

If all paths from the position $w_1 \notin W_X$ to any component positions of W_X pass through an intermediate position $w_3 \notin W_X$, then:

$$\pi(\Lambda \mid w_1) = \pi(\Lambda \mid w_3) \pi(w_3 \mid w_1)$$

Proof:

Consider a specific position $w \in W_X$, such that $w_1 \prec w$. Now, every path joining w_1 to w must pass through w_3 , so:

$$\begin{aligned} \pi(\Lambda(w) \mid \Lambda(w_1)) &= \frac{\pi(\Lambda(w_1, w))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(\Lambda(w_1, w_3, w))}{\pi(\Lambda(w_1))} \\ &= \pi(w_3, w \mid w_1) \\ &= \pi(w \mid w_3) \pi(w_3 \mid w_1) \end{aligned}$$

using Corollary 1 in section 3.6 and its proof. So:

$$\begin{aligned} \pi(\Lambda \mid w_1) &= \sum_{w \in W_X} \pi(\Lambda(w) \mid \Lambda(w_1)) \\ &= \sum_{w \in W_X} \pi(w \mid w_3) \pi(w_3 \mid w_1) \\ &= \pi(\Lambda \mid w_3) \pi(w_3 \mid w_1) \end{aligned} \quad \square$$

Now consider a position w_2 such that $w_1 \prec w_2 \prec w_3$. Since w_2 lies on a w_1 to W_X path and $w_2 \prec w_3$ then all w_2 to W_X paths must pass through w_3 . So $\pi(\Lambda \mid w_2) = \pi(\Lambda \mid w_3) \pi(w_3 \mid w_2)$.

Substituting into the expression given in Proposition 8, we get:

$$\begin{aligned} \hat{\pi}(w_2 \mid w_1) &= \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi(w_2 \mid w_1) \\ &= \frac{\pi(\Lambda \mid w_3) \pi(w_3 \mid w_2)}{\pi(\Lambda \mid w_3) \pi(w_3 \mid w_1)} \pi(w_2 \mid w_1) \\ &= \frac{\pi(w_3 \mid w_2)}{\pi(w_3 \mid w_1)} \pi(w_2 \mid w_1) \end{aligned}$$

This result is very useful, as when applying Algorithm step (5), if there is a position $w_3 \notin W_X$ such that all w_1 to W_X paths pass through w_3 , we can replace the updating formula given there by the one above and save a considerable amount of work:

Consider the abbreviated CEG in Figure 10, where $\Lambda = \Lambda(w_6)$, and all $w_1 \rightarrow w_6$ paths pass through w_5 .

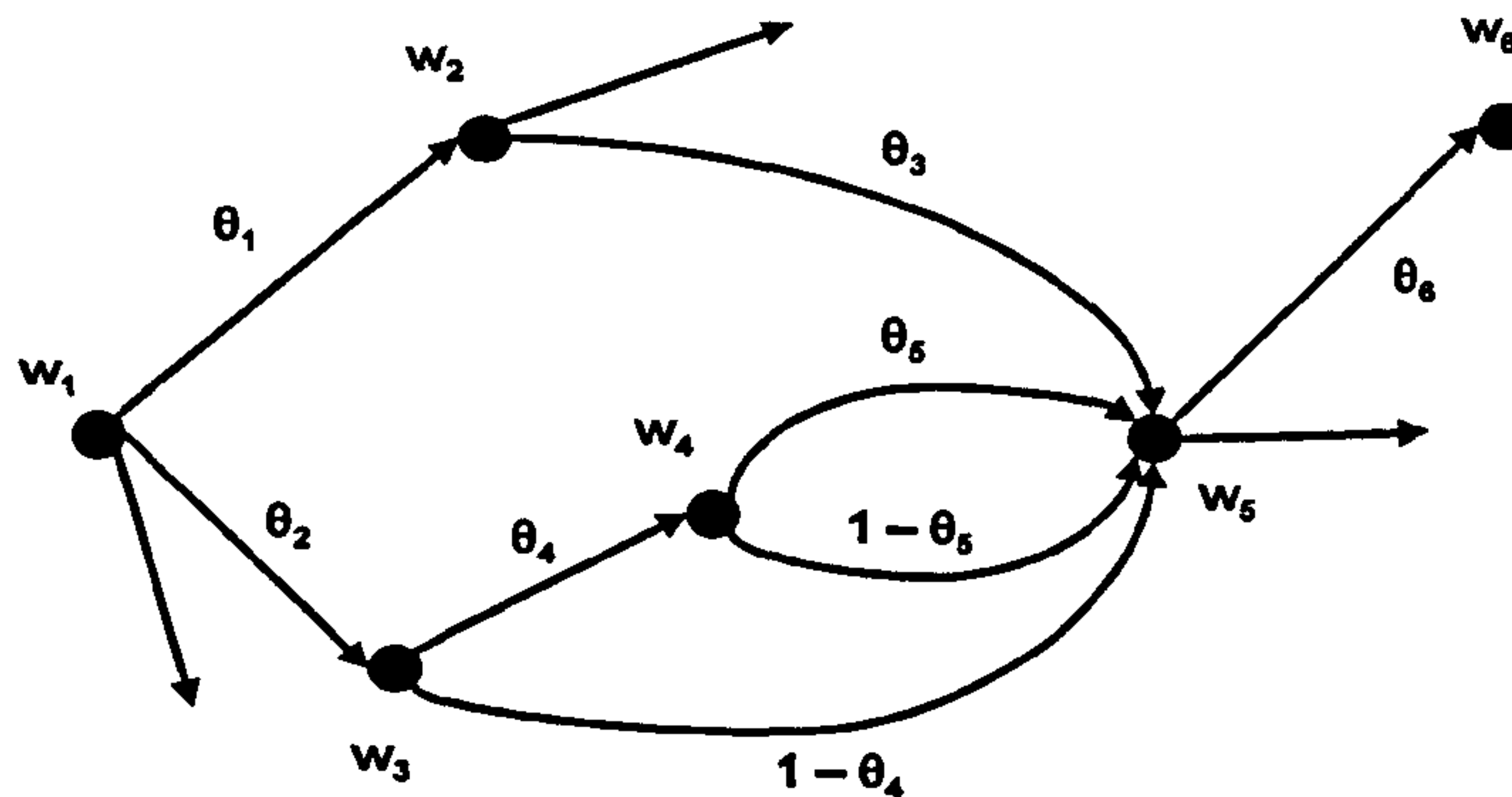


Figure 10: Abbreviated CEG

Here:

$$\begin{aligned}
 \hat{\pi}(w_2 \mid w_1) &= \frac{\pi(\Lambda \mid w_2)}{\pi(\Lambda \mid w_1)} \pi(w_2 \mid w_1) \\
 &= \frac{\theta_3 \theta_6}{(\theta_1 \theta_3 + \theta_2) \theta_6} \times \theta_1 \\
 &= \frac{\theta_3}{\theta_1 \theta_3 + \theta_2} \times \theta_1 \\
 &= \frac{\pi(w_5 \mid w_2)}{\pi(w_5 \mid w_1)} \pi(w_2 \mid w_1)
 \end{aligned}$$

The result's usefulness doesn't stop there. Suppose all paths leaving w_1 (not just all w_1 to W_X paths) pass through w_3 . Then:

$$\begin{aligned}
 \Lambda(w_1) &= \Lambda(w_1, w_3) \\
 \Rightarrow \pi(w_3 \mid w_1) &= \frac{\pi(\Lambda(w_1, w_3))}{\pi(\Lambda(w_1))} = 1
 \end{aligned}$$

Also, if a position w_2 is such that $w_1 \prec w_2 \prec w_3$, then all paths leaving w_2 must also pass through w_3 , so $\pi(w_3 \mid w_2) = 1$, and hence:

$$\begin{aligned}
 \hat{\pi}(w_2 \mid w_1) &= \frac{1}{1} \times \pi(w_2 \mid w_1) \\
 &= \pi(w_2 \mid w_1)
 \end{aligned}$$

Applying the result of Corollary 3 in this case, we get:

$$\begin{aligned}\hat{\pi}_e(w_2 \mid w_1) &= \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \hat{\pi}(w_2 \mid w_1) \\ &= \pi_e(w_2 \mid w_1)\end{aligned}$$

and by induction along each $w_1 \rightarrow w_3$ path-segment we get that the probabilities of all edges on all $w_1 \rightarrow w_3$ path-segments remain unchanged.

Suppose, in Figure 10, that all paths leaving w_3 pass through w_5 ; then:

$$\begin{aligned}\hat{\pi}_e(w_4 \mid w_3) &= \hat{\pi}(w_4 \mid w_3) \\ &= \frac{\pi(\Lambda \mid w_4)}{\pi(\Lambda \mid w_3)} \pi(w_4 \mid w_3) \\ &= \frac{\theta_6}{\theta_6} \times \theta_4 \\ &= \pi_e(w_4 \mid w_3) \\ \hat{\pi}(w_5 \mid w_4) &= \frac{\pi(\Lambda \mid w_5)}{\pi(\Lambda \mid w_4)} \pi(w_5 \mid w_4) \\ &= \frac{\theta_6}{\theta_6} \times 1 = 1\end{aligned}$$

so for the upper edge $e(w_4, w_5)$

$$\begin{aligned}\hat{\pi}_e(w_5 \mid w_4) &= \frac{\pi_e(w_5 \mid w_4)}{\pi(w_5 \mid w_4)} \hat{\pi}(w_5 \mid w_4) \\ &= \frac{\theta_5}{1} \times 1 \\ &= \pi_e(w_5 \mid w_4)\end{aligned}$$

Thus we can augment Algorithm step (5) as follows:

(5A) If all $w_1 \rightarrow W_X$ paths pass through w_3 , and $w_1 \prec w_2 \prec w_3$, $w_1 \sim w_2$, then:

$$\hat{\pi}(w_2 \mid w_1) = \frac{\pi(w_3 \mid w_2)}{\pi(w_3 \mid w_1)} \pi(w_2 \mid w_1)$$

and if there is more than one edge joining w_1 to w_2 , then:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\pi_e(w_2 \mid w_1)}{\pi(w_2 \mid w_1)} \hat{\pi}(w_2 \mid w_1)$$

(5B) If all paths leaving w_1 pass through w_3 , and $w_1 \prec w_2 \prec w_3$, $w_1 \sim w_2$, then:

$$\hat{\pi}(w_2 \mid w_1) = \pi(w_2 \mid w_1)$$

and if there is more than one edge joining w_1 to w_2 , then:

$$\hat{\pi}_e(w_2 \mid w_1) = \pi_e(w_2 \mid w_1)$$

(5C) If there is only one edge leaving w_1 which lies on a $w_1 \rightarrow W_X$ path, and if $w_1 \prec w_2 \prec w \in W_X$, $w_1 \sim w_2$, then:

$$\hat{\pi}_e(w_2 \mid w_1) = 1$$

In Figure 2 (from section 4.2) the only edge leaving w_2 which lies on a $w_2 \rightarrow w_6$ path is the edge $e(w_2, w_4)$, so step (5C) gives us that $\hat{\pi}_e(w_4 \mid w_2) = 1$, as we can see on Figure 6. Clearly this is quicker than using step (5) here.

4.5 Local Message Passing: Observation of a set of positions

It is sensible at this point to rewrite our algorithm so that it is more directly comparable with the Local Message Passing algorithms used for propagation on BNs. For ease of exposition I will exercise the flexibility of notation described in section 4.2 and let positions from $W(C)$ be labelled simply as w , positions downstream from but adjacent to a specified w will be labelled w' ; and where it is necessary to distinguish $w \in W_X$, such positions will be labelled w_a .

Local Message Passing Algorithm:

We follow the algorithm from section 4.2 for steps (1), (2) and (3), before introducing a variation which can be thought of as a Backward step followed by a Forward step (or as per [26] a *collecting* step and a *distributing* step).

- (1) Remove all undirected edges from the CEG
- (2) If a position $w \in W(C)$ ($w \notin W_X$) does not lie on a w_0 to $w_a \in W_X$ to w_∞ path then w and all edges entering or leaving w are pruned
- (3) All edges on path-segments $\mu(w_a, w_\infty)$ retain their probabilities from before, for all $w_a \in W_X$

Backward step:

- (4) For any $w \in W(C)$ remaining following step (2), let $W_w = \{w' \in W(C) \mid w' \succ w, w \sim w'\}$. Define the *emphasis* of w as:

$$\Phi(w) = \begin{cases} 0 & \text{for } w \notin W_X \\ 1 & \text{for } w \in W_X \\ \sum_{w' \in W_w} \pi(w' \mid w) \Phi(w') & \text{for } w \prec W_X \end{cases}$$

where $w \prec W_X$ means $w \prec w_a$ for some $w_a \in W_X$ etc.

- (4A) Considering only w in the set $\{w \in W(C) \mid w \prec w_a \text{ for some } w_a \in W_X\}$ (a subset of the set of w remaining following step (2)), replace each edge-probability $\pi_e(w' \mid w)$ by the *potential* $\tau_e(w' \mid w) = \pi_e(w' \mid w) \Phi(w')$.
- (4B) Repeat steps (4) and (4A) until all edges remaining upstream of W_X have been assigned a potential, and all positions remaining upstream of W_X have been assigned an emphasis.

Forward step:

- (5) Considering only w in the set $\{w \in W(C) \mid w \prec w_a \text{ for some } w_a \in W_X\}$, replace each potential $\tau_e(w' \mid w)$ by the updated edge-probability:

$$\hat{\pi}_e(w' \mid w) = \frac{\tau_e(w' \mid w)}{\Phi(w)}$$

We then follow step (6) from the algorithm in section 4.2:

- (6) In our updated graph any positions which are now equivalent are combined into a single position. Any positions which are now stage-equivalent are connected by an undirected edge, and edges are coloured as appropriate (see Definition 13).

We claim that $\hat{\pi}_e(w' \mid w)$ as assigned in step (5) above is the correct updated edge-probability. That is:

$$\hat{\pi}_e(w' \mid w) = \frac{\pi(\Lambda \mid w')}{\pi(\Lambda \mid w)} \pi_e(w' \mid w) \quad (\text{Corollary 3 expression (1)})$$

Before proceeding to a proof of this, I illustrate how it works by applying it to the example from section 4.3

Example 1 continued:

In this example I use our new LMP algorithm to update a selection of the edge-probabilities on the CEG in Figure 2 (in section 4.3) following observation of the event $\Lambda(w_6)$. The steps of the algorithm are not illustrated here by figures, but it should be noted that the process is very fast, and leads to the same CEG as given in Figures 8 and 9 in section 4.3.

First note that as $W_X = \{w_6\}$, we get $\Phi(w_6) = 1$.

Following the Backward steps (4) and (4A) we get:

$$\Phi(w_5) = \pi(w_6 \mid w_5) \Phi(w_6) = (\theta_9 + \theta_{10}) \times 1$$

$$\Phi(w_4) = \pi(w_6 \mid w_4) \Phi(w_6) = (\theta_6 + \theta_7) \times 1$$

$$\begin{aligned} \Phi(w_1) &= \pi(w_4 \mid w_1) \Phi(w_4) + \pi(w_5 \mid w_1) \Phi(w_5) \\ &= \theta_5(\theta_6 + \theta_7) + (1 - \theta_5)(\theta_9 + \theta_{10}) \end{aligned}$$

etc.

Note that as w_8 has been pruned in step (2) of the algorithm, we do not need to include in, for example, the expression for $\Phi(w_5)$, the term $\pi(w_8 \mid w_5)\Phi(w_8)$.

Also, for the **upper** edge $e(w_4, w_6)$

$$\begin{aligned} \tau_e(w_6 \mid w_4) &= \pi_e(w_6 \mid w_4) \Phi(w_6) \\ &= \theta_6 \times 1 = \theta_6 \end{aligned}$$

Similarly, for the **upper** edge $e(w_5, w_6)$

$$\tau_e(w_6 \mid w_5) = \pi_e(w_6 \mid w_5) \Phi(w_6) = \theta_9$$

and

$$\begin{aligned} \tau_e(w_4 \mid w_1) &= \pi_e(w_4 \mid w_1) \Phi(w_4) \\ &= \theta_5(\theta_6 + \theta_7) \end{aligned}$$

$$\begin{aligned} \tau_e(w_5 \mid w_1) &= \pi_e(w_5 \mid w_1) \Phi(w_5) \\ &= (1 - \theta_5)(\theta_9 + \theta_{10}) \end{aligned}$$

etc.

Applying the Forward step (5) to the edges leaving w_1 , we get:

$$\begin{aligned} \hat{\pi}_e(w_4 \mid w_1) &= \frac{\tau_e(w_4 \mid w_1)}{\Phi(w_1)} \\ &= \frac{\theta_5(\theta_6 + \theta_7)}{\theta_5(\theta_6 + \theta_7) + (1 - \theta_5)(\theta_9 + \theta_{10})} \quad \checkmark \end{aligned}$$

etc.

We now proceed to the proof of our Local Message Passing algorithm.

Proof of LMP algorithm:

We are claiming that $\hat{\pi}_e(w' \mid w)$ as assigned in step (5) of the algorithm is equal to $\frac{\pi(\Lambda \mid w')}{\pi(\Lambda \mid w)} \pi_e(w' \mid w)$.

But $\hat{\pi}_e(w' \mid w)$ as assigned in step (5) is equal to $\frac{\pi_e(w' \mid w) \Phi(w')}{\Phi(w)}$, so it is sufficient to show that $\Phi(w) = \pi(\Lambda \mid w)$ for all $w \in \{w \in W(C) \mid w \preceq w_a \text{ for some } w_a \in W_X\}$. We do this by induction:

Step 1.

Suppose $w \in W_X$. Then:

$$\begin{aligned}\pi(\Lambda \mid w) &= \frac{\pi(\Lambda, \Lambda(w))}{\pi(\Lambda(w))} \\ &= \frac{\pi(\bigcup_{w_a \in W_X} \Lambda(w_a), \Lambda(w))}{\pi(\Lambda(w))} \\ &= \frac{\pi(\Lambda(w))}{\pi(\Lambda(w))} = 1 = \Phi(w)\end{aligned}$$

from step (4) of the algorithm.

Step 2.

Suppose w ($w \prec W_X$) is such that $\Phi(w') = \pi(\Lambda \mid w') \quad \forall w' \in W_w$. Then:

$$\begin{aligned}\Phi(w) &= \sum_{w' \in W_w} \pi(w' \mid w) \Phi(w') \\ &= \sum_{w' \in W_w} \pi(w' \mid w) \pi(\Lambda \mid w') \\ &= \sum_{w' \in W_w} \pi(\Lambda(w') \mid \Lambda(w)) \pi(\Lambda \mid \Lambda(w')) \\ &= \sum_{w' \in W_w} \pi(\Lambda(w') \mid \Lambda(w)) \pi(\Lambda \mid \Lambda(w), \Lambda(w'))\end{aligned}$$

since $\Lambda = \bigcup_{w_a \in W_X} \Lambda(w_a)$ and $w \prec w' \prec w_a$ for some $w_a \in W_X$

[If $w \prec w' \prec w_a$ then $\pi(\Lambda(w_a) \mid \Lambda(w')) = \pi(\Lambda(w_a) \mid \Lambda(w), \Lambda(w'))$.

If $w \prec w' \not\prec w_a$ then $\pi(\Lambda(w_a) \mid \Lambda(w')) = 0 = \pi(\Lambda(w_a) \mid \Lambda(w), \Lambda(w'))$, so:

$$\begin{aligned}\pi(\Lambda \mid \Lambda(w')) &= \pi(\bigcup_{w_a \in W_X} \Lambda(w_a) \mid \Lambda(w')) \\ &= \sum_{w_a \in W_X} \pi(\Lambda(w_a) \mid \Lambda(w')) \\ &= \sum_{w_a \in W_X} \pi(\Lambda(w_a) \mid \Lambda(w), \Lambda(w')) \\ &= \pi(\Lambda \mid \Lambda(w), \Lambda(w')) \quad]\end{aligned}$$

Hence:

$$\begin{aligned}\Phi(w) &= \sum_{w' \in W_w} \pi(\Lambda, \Lambda(w') \mid \Lambda(w)) \\ &= \pi(\Lambda, \bigcup_{w' \in W_w} \Lambda(w') \mid \Lambda(w)) \\ &= \pi(\Lambda, \Lambda(w) \mid \Lambda(w)) \\ &= \pi(\Lambda \mid \Lambda(w)) \\ &= \pi(\Lambda \mid w)\end{aligned}$$

□

Now we know that Junction Tree algorithms for BNs are very efficient; and as our algorithm is directly analogous to these, this suggests that even for very complex CEGs our algorithm will be very quick. Note also that useful fast Junction Tree algorithms assume that observations need to be subsets of the variables depicted by the vertices of the Bayesian Network. This is not the case with a CEG, where our observation (in this section) can be of any event that can be expressed in the form $\bigcup_{w \in W_X} \Lambda(w)$ for some set W_X .

4.6 Propagation with general observation sets

In section 4.2 we constructed events of the form $\Lambda = \bigcup_{w \in W_X} \Lambda(w)$, and specified that no paths in our CEG should pass through more than one $w \in W_X$. At first sight we should have no real problems relaxing this latter condition, but consider the expression:

$$\begin{aligned} \pi(\Lambda \mid w_1) &= \pi\left(\bigcup_{w \in W_X} \Lambda(w) \mid \Lambda(w_1)\right) \\ &= \frac{\pi(\bigcup_{w \in W_X} \Lambda(w, w_1))}{\pi(\Lambda(w_1))} \end{aligned}$$

Now having relaxed the condition that $\Lambda(w_a, w_b) = \phi$ for $w_a, w_b \in W_X$, $w_a \neq w_b$, it is possible that we can find $w_1 \notin W_X$, $w_a, w_b \in W_X$, such that $w_1 \prec w_a \prec w_b$. Consider then:

$$\begin{aligned} \pi(\Lambda(w_1, w_a) \cup \Lambda(w_1, w_b)) &= \pi(\Lambda(w_1, w_a)) + \pi(\Lambda(w_1, w_b)) - \pi(\Lambda(w_1, w_a) \cap \Lambda(w_1, w_b)) \\ &= \pi(\Lambda(w_1, w_a)) + \pi(\Lambda(w_1, w_b)) - \pi(\Lambda(w_1, w_a, w_b)) \end{aligned}$$

In general therefore:

$$\begin{aligned} \pi\left(\bigcup_{w \in W_X} \Lambda(w, w_1)\right) &= \sum_{w \in W_X} \pi(\Lambda(w, w_1)) \\ &\quad - \sum_{w_i, w_j \in W_X, i \neq j} \pi(\Lambda(w_1, w_i, w_j)) \\ &\quad + \sum_{\substack{w_i, w_j, w_k \in W_X \\ i \neq j, k, j \neq k}} \pi(\Lambda(w_1, w_i, w_j, w_k)) \\ &\quad - \sum \dots \end{aligned}$$

where many, but not all of these terms may be zero.

This expression increases in complexity exponentially with the number of positions $w \in W_X$.

There may well be circumstances where the arithmetic is not onerous, but it does not really make sense to attempt to generalise the results of sections 4.2 to 4.5 to the case where we allow $\Lambda(w_a, w_b) \neq \phi$ for $w_a, w_b \in W_X$.

The set of events of the type $\Lambda = \bigcup_{w \in W_X} \Lambda(w)$ (where we may or may not impose the condition that $\Lambda(w_a, w_b) = \phi$, $w_a, w_b \in W_X$, $w_a \neq w_b$) is a restriction of the set of general events (ie. the path sigma-algebra of the CEG), where a general event can be characterised by $\Lambda = \bigcup_{i \in I} \lambda_i$ for some subset of the atomic events $\{\lambda_i\}$. We have hitherto considered propagation of probabilities following observation only of events that could be characterised as $\bigcup \Lambda(w)$ for some set of positions $\{w\}$. We now move on to develop methods for general observation sets of the form:

$$\Lambda = \bigcup_{i \in I} \lambda_i$$

where $\{\lambda_i\}$ are, as usual, the $w_0 \rightarrow w_\infty$ paths in our CEG.

If we follow the same approach as in section 4.2, we would get:

$$\begin{aligned} \pi(\Lambda \mid w_1) &= \pi\left(\bigcup_{i \in I} \lambda_i \mid \Lambda(w_1)\right) \\ &= \frac{\pi(\bigcup_{i \in I} \lambda_i, \Lambda(w_1))}{\pi(\Lambda(w_1))} \\ &= \frac{\pi(\bigcup_{i \in I} (\lambda_i, \Lambda(w_1)))}{\pi(\Lambda(w_1))} \end{aligned}$$

But $\bigcup_{i \in I} (\lambda_i, \Lambda(w_1))$ is simply those λ_j that are both elements of Λ and of $\Lambda(w_1)$, that is those λ_j that are elements of Λ and which pass through w_1 . These are clearly disjoint, so:

$$\begin{aligned} \pi(\Lambda \mid w_1) &= \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))}{\pi(\Lambda(w_1))} \\ &= \sum_{i \in I} \pi(\lambda_i \mid \Lambda(w_1)) \end{aligned}$$

If we continued with this approach, we would then attempt to produce expressions for $\hat{\pi}(w_1)$ ($= \pi(\Lambda(w_1) \mid \Lambda)$), $\hat{\pi}(w_2 \mid w_1)$, and finally for $\hat{\pi}_e(w_2 \mid w_1)$. Note that once we have defined $\Lambda = \bigcup_{i \in I} \lambda_i$, then we no longer have a special set of positions W_X . Positions labelled w_1, w_2, w_j, w_r etc. from hereon are simply positions from $W(C)$ which have been given the subscripts 1, 2, j , r etc. for notational convenience.

However, when we write $\hat{\pi}(w_2 \mid w_1)$ we are making the implicit assumption that the positions w_1 and w_2 exist in our CEG when we condition on Λ , and that they have the

same characteristics (such as histories) as in our unconditioned CEG. This is **not** a valid assumption for general observation sets Λ .

As we saw in section 4.2, conditioning on Λ actually alters the structure of the CEG, since edges are pruned, undirected edges may be permanently removed, and positions combined (as in the Blood Condition example). These types of alteration do not generally affect our ability to calculate post-observation probabilities, but it is none-the-less more accurate to say that the process is really the production of a new *conditioned CEG*, rather than simply the updating or propagation of probabilities on our original CEG.

This description of the process is given further weight when we see that there are some events which if conditioned upon radically alter the structure of the CEG by splitting positions (just as there are some events which if conditioned on alter the structure of a BN). If we condition on these types of events, we can no longer safely assume that if we can write $\pi(w_2 \mid w_1)$ in our unconditioned CEG, we can write $\hat{\pi}(w_2 \mid w_1)$ in our conditioned CEG and expect this to have a valid meaning, or a meaning that corresponds to that of the expression $\pi(w_2 \mid w_1)$.

Example 2:

Consider the CEG in Figure 11 which represents two independent binary variables A, B , each with outcome space $\{0, 1\}$. We could if we wished, envisage this as the tossing of two biased coins, with the variables taking the value 1 when the outcome of the toss is heads.

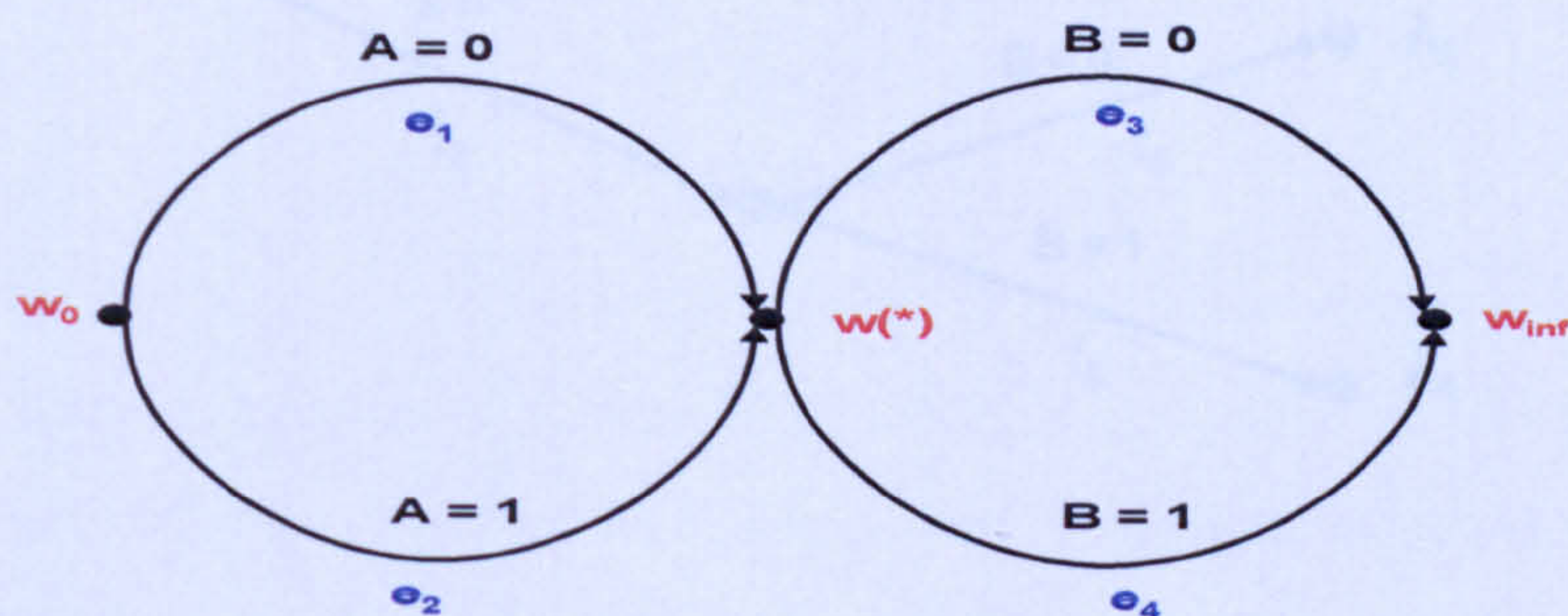


Figure 11: CEG for Example 2

Consider the event $\Lambda_1 = \{A = 0\}$ (the outcome when tossing the 1st coin is tails). Then the CEG conditioned upon Λ_1 (which we will call C_{Λ_1}) is given in Figure 12:

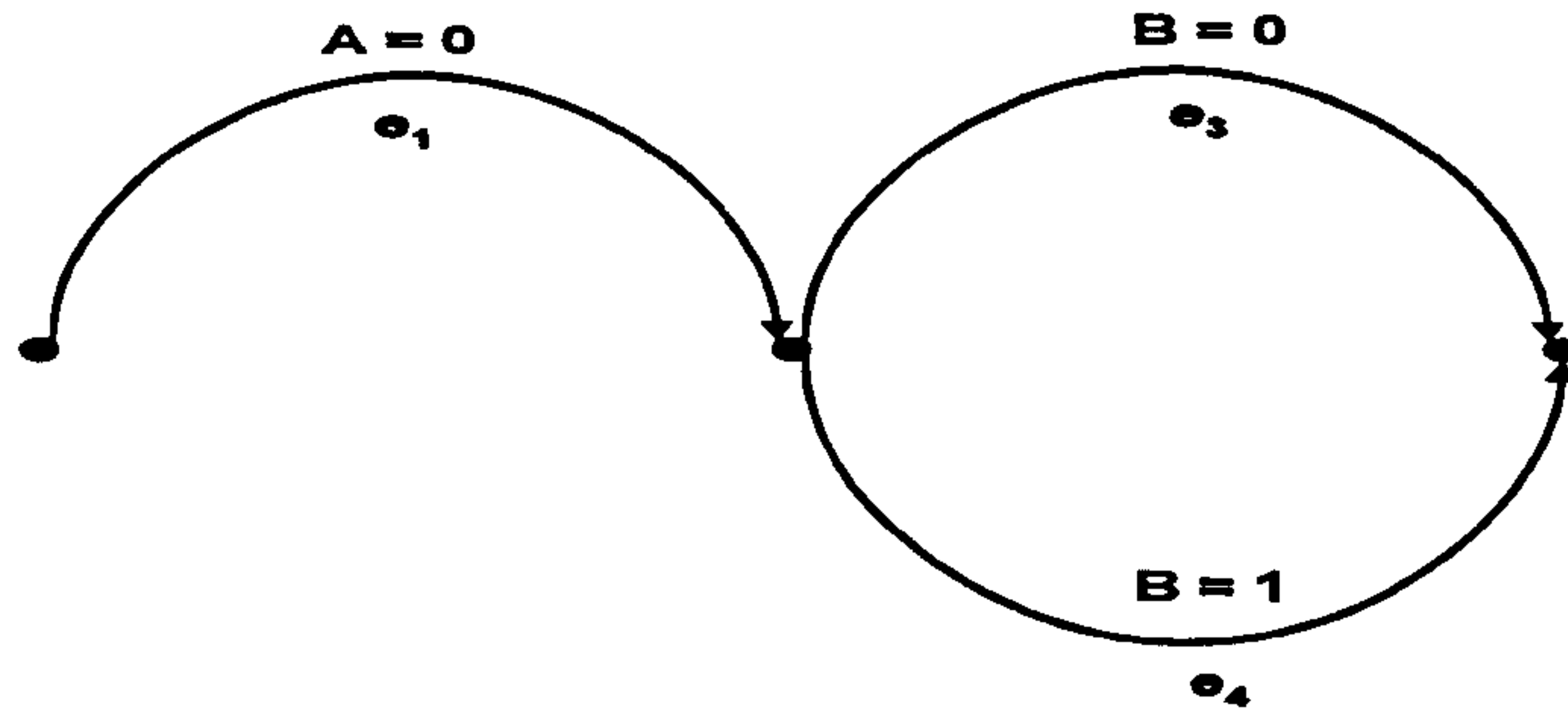


Figure 12: CEG conditioned on the event Λ_1

The edge probabilities in C_{Λ_1} are given by:

$$\hat{\pi}(e_1) = 1$$

$$\hat{\pi}(e_3) = \pi(B = 0 \mid A = 0)$$

$$= \pi(B = 0) = \pi(e_3) \quad \text{since } A \perp\!\!\!\perp B$$

$$\hat{\pi}(e_4) = \pi(B = 1) = \pi(e_4)$$

Consider now the event $\Lambda_2 = \{A = B\}$ (the outcomes when tossing the two coins are the same), and let us look at the underlying Tree:

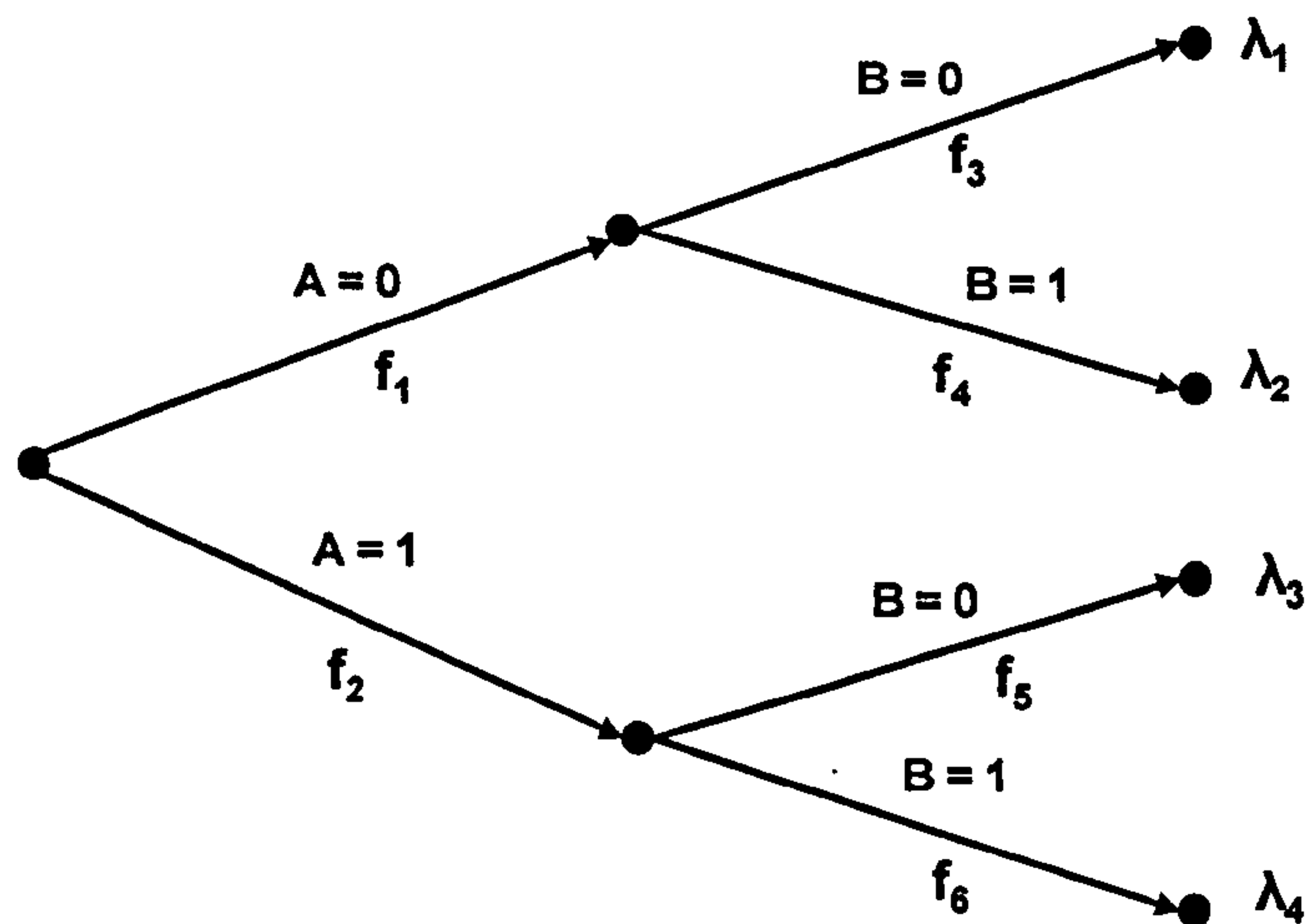


Figure 13: Tree showing the event Λ_2

Clearly $\Lambda_2 = \lambda_1 \cup \lambda_4$, and:

$$\begin{aligned} \hat{\pi}(f_1) &= \frac{\pi(A = 0, B = 0)}{\pi((A = 0, B = 0) \cup (A = 1, B = 1))} \\ &= \frac{\pi(A = 0) \pi(B = 0)}{\pi(A = 0) \pi(B = 0) + \pi(A = 1) \pi(B = 1)} \end{aligned}$$

$$= \frac{\pi(e_1) \pi(e_3)}{\pi(e_1) \pi(e_3) + \pi(e_2) \pi(e_4)}$$

since $A \amalg B$. Also:

$$\hat{\pi}(f_3) = \hat{\pi}(f_6) = 1$$

Our conditioned CEG C_{Λ_2} is given in Figure 14, and we can see that conditioning on Λ_2 has split the position $w(*)$ into two positions $w(0)$ and $w(1)$. The reason for this is that in our original CEG, we had $A \amalg B$, but we do **not** have $A \amalg B \mid \Lambda_2$. Conditioning on an event Λ will split positions if independence relationships in our original CEG do not hold when conditioned on Λ .

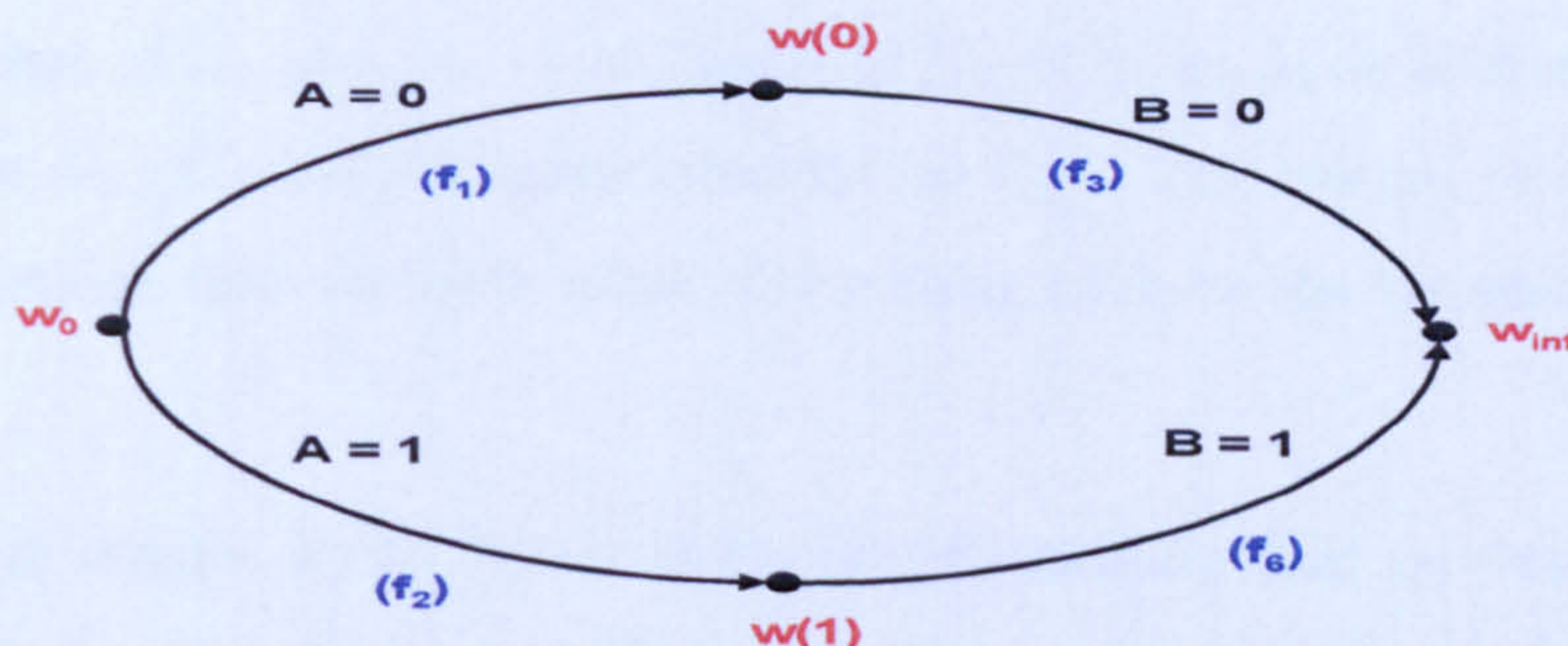


Figure 14: CEG conditioned on the event Λ_2

Definition 25: For a CEG $C(W(C), E_d(C), E_u(C))$, and event $\Lambda = \bigcup_{i \in I} \lambda_i$, let $G_\Lambda(C)(V(G_\Lambda(C)), E(G_\Lambda(C)))$ be the subgraph of C with:

- (a) $V(G_\Lambda(C)) \subset W(C)$ contains exactly those vertices (positions in C) which lie on a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
- (b) $E(G_\Lambda(C)) \subset E_d(C)$ contains exactly those edges which form part of a path $\lambda_i \in \{\lambda_i\}_{i \in I}$ [Note that $E(G_\Lambda(C)) \cap E_u(C) = \emptyset$]
- (c) Edges in $E(G_\Lambda(C))$ have **no** probability labels.

We will call this graph the subgraph (of C) defined by Λ , and normally denote it by $G_\Lambda(C)$. Note that it can be obtained from C by (i) removing all **probability** labels and undirected edges, (ii) retaining all positions and **directed** edges consistent with Λ (ie. lying on some path $\lambda \in \Lambda$), and removing all other positions and edges. $G_\Lambda(C)$ is not normally a CEG.

Definition 26: An event $\Lambda = \bigcup_{i \in I} \lambda_i$ is *intrinsic* to a CEG C if:

- (i) Every λ_i ($i \in I$) is a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$, the subgraph defined by Λ ,
- (ii) Every $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$ is a λ_i for some $i \in I$.

So for example, for the CEG in Figure 11, the subgraph defined by Λ_1 is given in Figure 15. Note that it contains two $w_0 \rightarrow w_\infty$ paths, each of which is an element of Λ_1 .

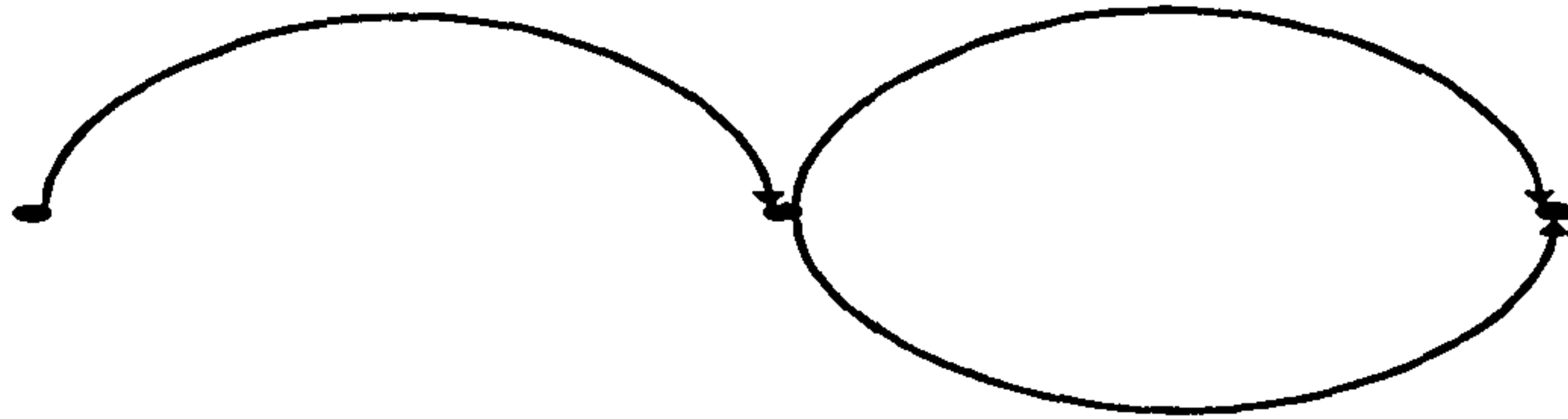


Figure 15: Subgraph of C defined by the event Λ_1

Also, each element of Λ_1 is a $w_0 \rightarrow w_\infty$ path in $G_{\Lambda_1}(C)$, so Λ_1 is intrinsic to C . Note that in this case $G_{\Lambda_1}(C)$ has the same structure as C_{Λ_1} . The two $w_0 \rightarrow w_\infty$ paths here correspond to getting tails on both coins; and getting tails on the 1st coin, heads on the 2nd coin.

But the subgraph defined by Λ_2 (given in Figure 16) contains four $w_0 \rightarrow w_\infty$ paths, only two of which $((A = 0, B = 0), (A = 1, B = 1)$ or (tails, tails), (heads, heads)) are elements of Λ_2 , so Λ_2 is not intrinsic to C . Note that $G_{\Lambda_2}(C)$ has a different structure to C_{Λ_2} .

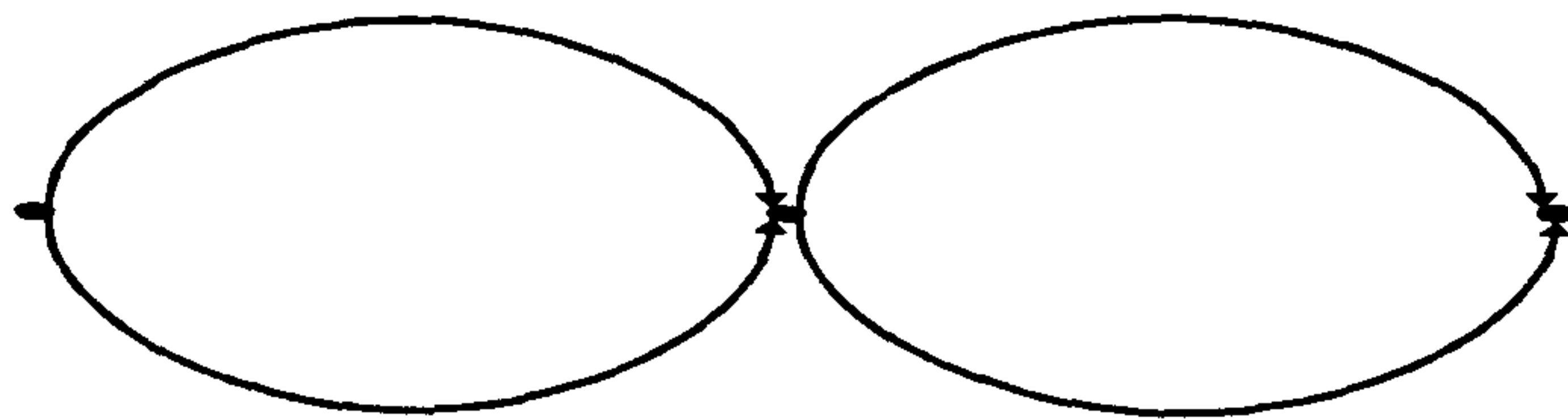


Figure 16: Subgraph of C defined by the event Λ_2

It is worth at this point spending a little time considering conditioning on BNs. Typically we would not concern ourselves with the complete conditional independence structure of the problem, as our observations will be of subsets of the measurement variables, and the triangularisation process inherent in BN-based propagation removes those conditional independence statements that are not closed under sampling. This simplification of the process should be reflected in CEG-based propagation.

But for many processes, the variables one can measure (the BN's vertices) are not the underlying variables on which the process functions. Hence a BN based on the measurement variables may not adequately describe the underlying conditional independence structure of the process; and so the analyst may need to transform the BN's variables before he/she can perform propagation (or indeed any analysis) upon the BN.

In a CEG we have much more flexibility. Firstly, the CEG is a description of the process, so any variables we create may well be more appropriate than the measurement variables of the BN. Secondly, the fact that we can create variables is obviously an advantage over having to use variables provided. Thirdly, when we construct our CEG we can consider not only what conditional independence structure a problem has before we observe an event, but also how the topology of the graph might be affected by potential observations. By only encoding those conditional independence statements that will not be destroyed by our observations, we can ensure that the observations which we attempt to propagate are intrinsic to our CEG. This will be very useful as will be seen later in this section.

Note that not encoding some of our conditional independence statements will produce a more Tree-like CEG, with fewer positions with more than one incoming edge, and fewer positions joined by undirected edges.

Returning to the development of our toolbox for propagation with general observation sets, we now define C_Λ , the CEG conditioned upon the event Λ , but we do not do this from the subgraph $G_\Lambda(C)$. Instead we note that our CEG C is a function of a Probability Tree T (section 2.5), and use this fact to define our C_Λ . We start by creating T_Λ , the Tree conditioned upon Λ :

Recall that Λ has been defined on our CEG as a union of $w_0 \rightarrow w_\infty$ paths ($\Lambda = \bigcup_{i \in I} \lambda_i$). However, there is a one-to-one correspondence between these paths and root-to-leaf paths in the Tree T . When we define Λ on T it is therefore totally unambiguous to write $\Lambda = \bigcup_{i \in I} \lambda_i$ where $\{\lambda_j\}$ is the set of root-to-leaf paths in T , and $\{\lambda_i\}_{i \in I}$ is the set of paths in T corresponding to the $w_0 \rightarrow w_\infty$ paths in Λ defined on C .

Definition 27: For a Tree $T(V(T), E(T))$, and an event Λ , let $T_\Lambda(V(T_\Lambda), E(T_\Lambda))$ be the subgraph (subtree) of T with:

- (a) $V(T_\Lambda) \subset V(T)$ contains exactly those vertices which lie on a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
- (b) $E(T_\Lambda) \subset E(T)$ contains exactly those edges which form part of a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
- (c) Probabilities in T_Λ (denoted by $\hat{\pi}$) are governed by the formula:

$$\hat{\pi}(\lambda_i) = \frac{\pi(\lambda_i)}{\pi(\Lambda)}$$

Hence, for $v_r, v_s \in V(T_\Lambda)$ such that there exists an edge $(e(v_r, v_s)) \in E(T_\Lambda)$:

$$\hat{\pi}_e(v_s \mid v_r) = \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(v_r, v_s)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(v_r))}$$

Note that T_Λ can be obtained from T by (i) removing any edges that do not form part of some λ_i for $i \in I$, (ii) replacing all edge probabilities $\pi(e_j)$ by $\hat{\pi}(e_j)$ in accordance with (c) above.

Definition 28: Define C_Λ to be the CEG derived from the Tree T_Λ (see Definitions 9 and 13).

Proposition 11:

If Λ is intrinsic to C then C_Λ can be obtained from C by a sequence of:

- (1) pruning directed edges and positions,
- (2) combining positions,
- (3) removal and addition of undirected edges.

(ie. by a process similar to that described in the algorithm in section 4.2)

If Λ is not intrinsic to C then C_Λ cannot be so obtained. C_Λ will contain positions not present in C .

We might choose to describe this property as *Intrinsic events respect positions*, but this is probably an oversimplification.

Proof (part 1)

Suppose C_Λ can not be obtained from C by a sequence of:

- (1) pruning directed edges and positions,
- (2) combining positions,
- (3) removal and addition of undirected edges.

Then C_Λ must contain either additional directed edges or additional positions or both.

Now consider corresponding positions w_i in C and C_Λ , and the edges emanating from these positions. There cannot be more edges emanating from w_i in C_Λ than emanating from w_i in C , since this would require that the outcome space of X_i in C_Λ was larger than that of X_i in C , which is impossible given the derivation of C_Λ from the Tree T_Λ , which is a pruned version of the Tree T .

So if C_Λ cannot be obtained from C by a sequence of pruning directed edges and positions, combining positions, and removal/addition of undirected edges, then there must exist additional positions in C_Λ , which can only have been produced by not combining positions in T when producing C_Λ which would have been combined when producing C . That is, there are position(s) in C that are *split* in C_Λ .

Consider a position w in C corresponding to a set of vertices in T . Without loss of generality, let this set of vertices in T correspond to two distinct positions w_r and w_s in C_Λ . Let v_i be a representative of the vertices in T corresponding to w_r , and v_j be a representative corresponding to w_s . Then, since w_r and w_s are distinct in C_Λ , we must have:

$$Y_s \not\equiv Y_t \quad \text{in } C_\Lambda$$

Note that as Y_k etc. (Definition 5 in section 2.3 and Definition 10 in section 2.5) are defined on a specific CEG or Tree, there is no ambiguity in making this statement.

Hence:

$$Y_i \not\equiv Y_j \quad \text{in } T_\Lambda$$

and either:

- (i) Y_i and Y_j must have different outcome spaces in T_Λ , or
- (ii) Y_i and Y_j have the same outcome spaces in T_Λ , but have different probability distributions over these spaces.

Consider case (ii):

Then for every element of the outcome space of Y_i in T_Λ there is a corresponding element in the outcome space of Y_j in T_Λ . As these elements correspond to $v_i \rightarrow \text{leaf}$ or $v_j \rightarrow \text{leaf}$ path-segments, we will use the symbols μ_i^s and μ_j^s to represent these elements.

Now $Y_i \equiv Y_j$ in T (since the vertices v_i and v_j both correspond to the position w in C). So in T :

$$\pi_{Y_i}(\mu_i^s) = \pi_{Y_j}(\mu_j^s)$$

for corresponding μ_i^s and μ_j^s . Hence, in T :

$$\pi(\mu_i^s \mid \Lambda(v_i)) = \pi(\mu_j^s \mid \Lambda(v_j))$$

Now, as we are working with Trees we can clearly write this as:

$$\frac{\pi(\Lambda(v_i, \mu_i^s))}{\pi(\Lambda(v_i))} = \frac{\pi(\Lambda(v_j, \mu_j^s))}{\pi(\Lambda(v_j))}$$

where $\Lambda(v_i, \mu_i^s)$ is the event which is the set of paths passing through v_i and utilising the path-segment μ_i^s . But we are working on a Tree so this is a single path, which we can denote λ_i^s , and we can write:

$$\frac{\pi(\lambda_i^s)}{\pi(\Lambda(v_i))} = \frac{\pi(\lambda_j^s)}{\pi(\Lambda(v_j))}$$

If we now denote probabilities in T_Λ by $\hat{\pi}$, we get, using (C) of the definition of T_Λ (Definition 27):

$$\begin{aligned}
& \frac{\pi(\Lambda) \hat{\pi}(\lambda_i^s)}{\pi(\Lambda(v_i))} = \frac{\pi(\Lambda) \hat{\pi}(\lambda_j^s)}{\pi(\Lambda(v_j))} \\
& \Rightarrow \frac{\hat{\pi}(\lambda_i^s)}{\pi(\Lambda(v_i))} = \frac{\hat{\pi}(\lambda_j^s)}{\pi(\Lambda(v_j))} \\
& \Rightarrow \frac{\hat{\pi}(\Lambda(v_i, \mu_i^s))}{\pi(\Lambda(v_i))} = \frac{\hat{\pi}(\Lambda(v_j, \mu_j^s))}{\pi(\Lambda(v_j))} \\
& \Rightarrow \hat{\pi}(\mu_i^s \mid \Lambda(v_i)) \frac{\hat{\pi}(\Lambda(v_i))}{\pi(\Lambda(v_i))} = \hat{\pi}(\mu_j^s \mid \Lambda(v_j)) \frac{\hat{\pi}(\Lambda(v_j))}{\pi(\Lambda(v_j))}
\end{aligned}$$

But $\pi(\Lambda(v_i))$, $\hat{\pi}(\Lambda(v_i))$ etc. are constants, so we get:

$$\begin{aligned}
& \hat{\pi}(\mu_i^s \mid \Lambda(v_i)) = k \hat{\pi}(\mu_j^s \mid \Lambda(v_j)) \\
& \Rightarrow \hat{\pi}_{Y_i}(\mu_i^s) = k \hat{\pi}_{Y_j}(\mu_j^s) \\
& \Rightarrow 1 = \sum_s \hat{\pi}_{Y_i}(\mu_i^s) = k \sum_s \hat{\pi}_{Y_j}(\mu_j^s) = k \\
& \Rightarrow \hat{\pi}_{Y_i}(\mu_i^s) = \hat{\pi}_{Y_j}(\mu_j^s) \quad \forall s \\
& \Rightarrow Y_i \equiv Y_j \quad \text{in } T_\Lambda
\end{aligned}$$

which contradicts our premise.

So case (ii) is impossible, and we must have case (i): Y_i and Y_j have different outcome spaces in T_Λ .

Without loss of generality, let there exist one $v_i \rightarrow$ leaf path-segment in the outcome space of Y_i in T_Λ that has **no** corresponding $v_j \rightarrow$ leaf path-segment in the outcome space of Y_j in T_Λ .

Now v_i corresponds to $w \in C$, so in $G_\Lambda(C)$ there must exist a $w \rightarrow w_\infty$ path-segment corresponding to this $v_i \rightarrow$ leaf path-segment.

But v_j also corresponds to $w \in C$, so in $G_\Lambda(C)$ there will be a $w_0 \rightarrow w_\infty$ path that consists of a $w_0 \rightarrow w$ path-segment corresponding to the $v_0 \rightarrow v_j$ path-segment in T conjoined to the above $w \rightarrow w_\infty$ path-segment. This $w_0 \rightarrow w_\infty$ path corresponds to **no** $v_0 \rightarrow$ leaf path in T_Λ .

Hence there exists a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$ which is **not** an element of $\{\lambda_i\}_{i \in I}$.

Hence Λ is not intrinsic to C .

Proof (part 2)

Suppose Λ is not intrinsic to C . Then either:

- (i) There is a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$ which is not an element of $\{\lambda_i\}_{i \in I}$, or
- (ii) There is an element of $\{\lambda_i\}_{i \in I}$ which is not a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$.

Now case (ii) is impossible by (b) of the definition of $G_\Lambda(C)$ (Definition 25). Hence there exists a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$ which is not an element of $\{\lambda_i\}_{i \in I}$.

By (b) of the definition of $G_\Lambda(C)$, this path must consist totally of edges which are utilised by some element(s) of $\{\lambda_i\}_{i \in I}$. Therefore there must exist a $w_0 \rightarrow w_\infty$ path in C which utilises only Λ -consistent edges (edges which are part of some $w_0 \rightarrow w_\infty$ path $\lambda_i \in \Lambda$), but which is itself not an element of Λ .

Can we obtain C_Λ from C by a sequence of:

- (1) pruning directed edges and positions,
- (2) combining positions,
- (3) removal and addition of undirected edges?

To be able to answer Yes to this question we must be able to remove this $w_0 \rightarrow w_\infty$ path from C by such a sequence, since this path cannot exist in C_Λ . So consider each aspect of the process in turn:

- (3) Our $w_0 \rightarrow w_\infty$ path does not utilise any undirected edges, so removal or addition of such edges will not help us to remove our $w_0 \rightarrow w_\infty$ path,
- (1) All edges on our $w_0 \rightarrow w_\infty$ path are used by elements of Λ , so we cannot prune them. Similarly all positions on our path separate edges used by elements of Λ , so are themselves needed by elements of Λ , and so we cannot prune them,
- (2) Combining positions cannot decrease the number of $w_0 \rightarrow w_\infty$ paths in a CEG (or replace $w_0 \rightarrow w_\infty$ paths by others), so we cannot remove our $w_0 \rightarrow w_\infty$ path by doing this. Neither can we combine combining of positions with pruning some edges entering or leaving these positions, since the edges will still be required by elements of Λ .

Therefore, we cannot remove our $w_0 \rightarrow w_\infty$ path by a sequence (1),(2),(3). Therefore we cannot obtain C_Λ from C by a sequence (1),(2),(3).

□

Note: We have already shown that if we cannot obtain C_Λ from C by a sequence (1),(2),(3), then C_Λ has *additional* positions which are the result of *splitting* positions in C .

What does this mean?

It means that when we observe Λ (or condition on Λ), what we actually need to do is produce a conditioned-CEG C_Λ . If and only if Λ is intrinsic to C , can we produce C_Λ directly from C by a sequence (1),(2),(3), rather than having to go back to the original Tree.

So, if Λ is intrinsic to C , we can create an algorithm which allows us to obtain C_Λ from C in a way which could be described as *updating C following observation* or *propagating probabilities on C* .

4.7 An algorithm for propagation following observation of intrinsic events

Proposition 12:

For an event $\Lambda = \bigcup_{i \in I} \lambda_i$, intrinsic to a CEG C , and where the expression $\hat{\pi}_e(w_2 | w_1)$ (the edge-probability of the edge $e(w_1, w_2)$ in the conditioned CEG C_Λ) has a valid meaning:

$$\hat{\pi}_e(w_2 | w_1) = \frac{\pi(\Lambda | \Lambda(e(w_1, w_2)))}{\pi(\Lambda | \Lambda(w_1))} \pi_e(w_2 | w_1) \quad (A)$$

$$= \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))} \quad (B)$$

Note that the positions w_1, w_2 in the expression $\hat{\pi}_e(w_2 | w_1)$ will be those in C_Λ that correspond to the positions w_1, w_2 in C - they may **not** have the same labels!

Proof:

$$\pi_e(w_2 | w_1) = \pi(\Lambda(e(w_1, w_2)) | \Lambda(w_1)) \quad (\text{Result 3 from section 3.6})$$

So:

$$\begin{aligned} \hat{\pi}_e(w_2 | w_1) &= \pi(\Lambda(e(w_1, w_2)) | \Lambda, \Lambda(w_1)) \\ &= \frac{\pi(\Lambda, \Lambda(w_1), \Lambda(e(w_1, w_2)))}{\pi(\Lambda, \Lambda(w_1))} \\ &= \frac{\pi(\Lambda | \Lambda(w_1), \Lambda(e(w_1, w_2))) \pi(\Lambda(e(w_1, w_2)) | \Lambda(w_1))}{\pi(\Lambda | \Lambda(w_1))} \end{aligned} \quad (4.7.1)$$

But $\Lambda(e(w_1, w_2)) \subset \Lambda(w_1)$, so $\Lambda(w_1) \cap \Lambda(e(w_1, w_2)) = \Lambda(e(w_1, w_2))$, implying that $\pi(\Lambda | \Lambda(w_1), \Lambda(e(w_1, w_2))) = \pi(\Lambda | \Lambda(e(w_1, w_2)))$ and hence:

$$\hat{\pi}_e(w_2 | w_1) = \frac{\pi(\Lambda | \Lambda(e(w_1, w_2)))}{\pi(\Lambda | \Lambda(w_1))} \pi_e(w_2 | w_1) \quad (A)$$

We have already noted that $\Lambda(e(w_1, w_2)) \subset \Lambda(w_1)$. One consequence of this is that $\Lambda \cap \Lambda(w_1) \cap \Lambda(e(w_1, w_2)) = \Lambda \cap \Lambda(e(w_1, w_2))$, and hence we can rewrite expression (4.7.1)

as:

$$\begin{aligned}\hat{\pi}_e(w_2 \mid w_1) &= \frac{\pi(\Lambda, \Lambda(e(w_1, w_2)))}{\pi(\Lambda, \Lambda(w_1))} \\ &= \frac{\pi(\bigcup_{i \in I} \lambda_i, \Lambda(e(w_1, w_2)))}{\pi(\bigcup_{i \in I} \lambda_i, \Lambda(w_1))} \\ &= \frac{\pi(\bigcup_{i \in I} (\lambda_i, \Lambda(e(w_1, w_2))))}{\pi(\bigcup_{i \in I} (\lambda_i, \Lambda(w_1)))}\end{aligned}$$

since the events $\{\lambda_i\}$ are disjoint

$$= \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))} \quad (B)$$

since the events $\{(\lambda_i, \Lambda(e(w_1, w_2)))\}$ and $\{(\lambda_i, \Lambda(w_1))\}$ are disjoint.

□

Note that if λ_k does not utilise the edge $e(w_1, w_2)$ then $\pi(\lambda_k, \Lambda(e(w_1, w_2))) = 0$. Similarly, if λ_k does not pass through w_1 then $\pi(\lambda_k, \Lambda(w_1)) = 0$. Hence $\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))$ is simply summing the probabilities of those $\lambda_j \in \Lambda$ that utilise the edge $e(w_1, w_2)$, and $\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))$ is simply summing the probabilities of those $\lambda_j \in \Lambda$ that pass through w_1 .

These facts make both the construction and use of an algorithm for propagation following observation of Intrinsic events very straightforward.

Algorithm

Following our observation of the event $\Lambda = \bigcup_{i \in I} \lambda_i$, intrinsic to the CEG C , we construct C_Λ , the CEG conditioned on the event Λ , as follows:

- (1) Remove all undirected edges from the CEG
- (2) All positions and edges which do not lie on a $w_0 \rightarrow w_\infty$ path $\lambda_i \in \{\lambda_i\}_{i \in I}$, are pruned
- (3) Each remaining edge is updated as:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))}$$

- (4) In our updated graph any positions which are now equivalent are combined into a single position. Any positions which are now stage-equivalent are connected by an undirected edge, and edges are coloured as appropriate (see Definition 13).

Note that if all $\lambda_i \in \{\lambda_i\}_{i \in I}$ which pass through w_1 , also pass through $e(w_1, w_2)$, then we have $\Lambda \cap \Lambda(w_1) = \Lambda \cap \Lambda(e(w_1, w_2))$ and hence $\hat{\pi}_e(w_2 \mid w_1) = 1$.

Example 3: Another look at the Blood condition example

Returning to the Blood Condition example from section 4.3, and considering the observation that a patient is in group yr or zr , can we use our new algorithm to construct a CEG conditioned on this event?

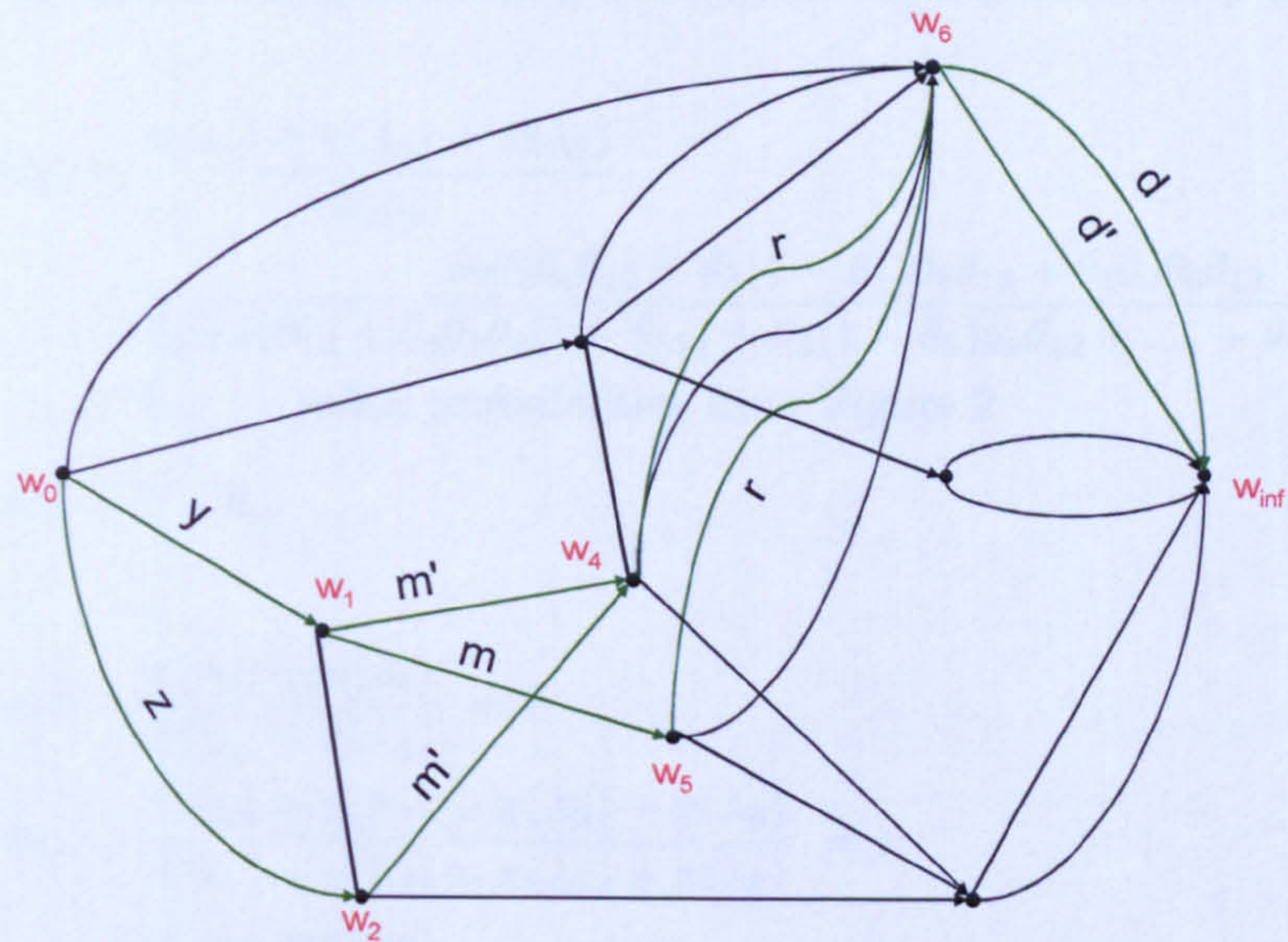


Figure 17: CEG showing component paths of Λ

Now $\Lambda = \bigcup_{i=1}^6 \lambda_i$, where:

$$\begin{array}{lll} \lambda_1 = ym'rd & \lambda_3 = ymrd & \lambda_5 = zm'rd \\ \lambda_2 = ym'rd' & \lambda_4 = ymrd' & \lambda_6 = zm'rd' \end{array}$$

The subgraph $G_\Lambda(C)$ defined by Λ is given in Figure 18 (with the addition of edge numbers to aid the analysis):

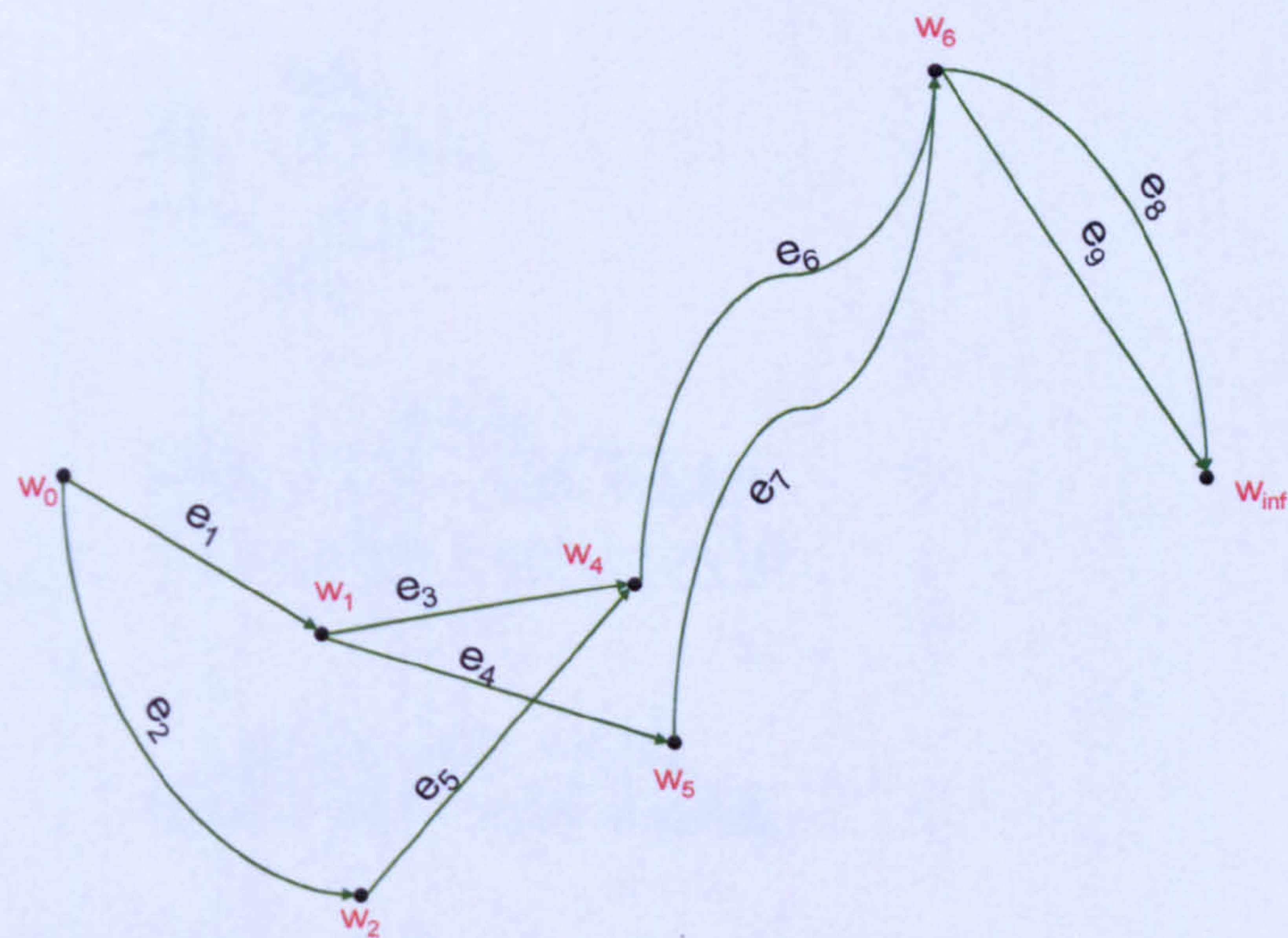


Figure 18: The subgraph $G_\Lambda(C)$

Now every $\lambda_i \in \Lambda$ is a $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$, and every $w_0 \rightarrow w_\infty$ path in $G_\Lambda(C)$ is a $\lambda_i \in \Lambda$, so Λ is intrinsic to C , and we can use our algorithm.

Algorithm steps (1) and (2) give the graph in Figure 18. Algorithm step (3) gives:

$$\begin{aligned}\hat{\pi}_{e_8}(w_\infty \mid w_6) &= \frac{\pi(\lambda_1) + \pi(\lambda_3) + \pi(\lambda_5)}{\pi(\Lambda)} \\ &= \frac{\theta_3\theta_5\theta_6\theta_{12} + \theta_3(1 - \theta_5)\theta_9\theta_{12} + \theta_4\theta_5\theta_6\theta_{12}}{\theta_3\theta_5\theta_6\theta_{12} + \theta_3\theta_5\theta_6(1 - \theta_{12}) + \theta_3(1 - \theta_5)\theta_9\theta_{12} + \dots + \theta_4\theta_5\theta_6\theta_{12} + \dots} \\ &= \theta_{12} \quad \text{using probabilities from Figure 2}\end{aligned}$$

$$\Rightarrow \hat{\pi}_{e_9}(w_\infty \mid w_6) = 1 - \theta_{12}$$

Also:

$$\begin{aligned}\hat{\pi}_{e_7}(w_6 \mid w_5) &= \frac{\pi(\lambda_3) + \pi(\lambda_4)}{\pi(\lambda_3) + \pi(\lambda_4)} = 1 \\ \hat{\pi}_{e_6}(w_6 \mid w_4) &= \frac{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_5) + \pi(\lambda_6)}{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_5) + \pi(\lambda_6)} = 1 \\ \hat{\pi}_{e_5}(w_4 \mid w_2) &= \frac{\pi(\lambda_5) + \pi(\lambda_6)}{\pi(\lambda_5) + \pi(\lambda_6)} = 1 \\ \hat{\pi}_{e_4}(w_5 \mid w_1) &= \frac{\pi(\lambda_3) + \pi(\lambda_4)}{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3) + \pi(\lambda_4)} \\ &= \frac{\theta_3(1 - \theta_5)\theta_9\theta_{12} + \theta_3(1 - \theta_5)\theta_9(1 - \theta_{12})}{\theta_3\theta_5\theta_6\theta_{12} + \theta_3\theta_5\theta_6(1 - \theta_{12}) + \theta_3(1 - \theta_5)\theta_9\theta_{12} + \theta_3(1 - \theta_5)\theta_9(1 - \theta_{12})} \\ &= \frac{(1 - \theta_5)\theta_9}{\theta_5\theta_6 + (1 - \theta_5)\theta_9} \\ \hat{\pi}_{e_3}(w_4 \mid w_1) &= \frac{\pi(\lambda_1) + \pi(\lambda_2)}{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3) + \pi(\lambda_4)} \\ &= \dots \\ &= \frac{\theta_5\theta_6}{\theta_5\theta_6 + (1 - \theta_5)\theta_9} \\ \hat{\pi}_{e_2}(w_2 \mid w_0) &= \frac{\pi(\lambda_5) + \pi(\lambda_6)}{\pi(\Lambda)} \\ &= \dots \\ &= \frac{\theta_4\theta_5\theta_6}{\theta_3\theta_5\theta_6 + \theta_3(1 - \theta_5)\theta_9 + \theta_4\theta_5\theta_6} \\ \hat{\pi}_{e_1}(w_1 \mid w_0) &= \frac{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3) + \pi(\lambda_4)}{\pi(\Lambda)} \\ &= \dots \\ &= \frac{\theta_3\theta_5\theta_6 + \theta_3(1 - \theta_5)\theta_9}{\theta_3\theta_5\theta_6 + \theta_3(1 - \theta_5)\theta_9 + \theta_4\theta_5\theta_6}\end{aligned}$$

The graph following algorithm step (3) is given in Figure 19 (where $\phi_3 = \hat{\pi}_{e_1}(w_1 \mid w_0)$, $\phi_4 = \hat{\pi}_{e_2}(w_2 \mid w_0)$, and $\phi_5 = \hat{\pi}_{e_3}(w_4 \mid w_1)$):

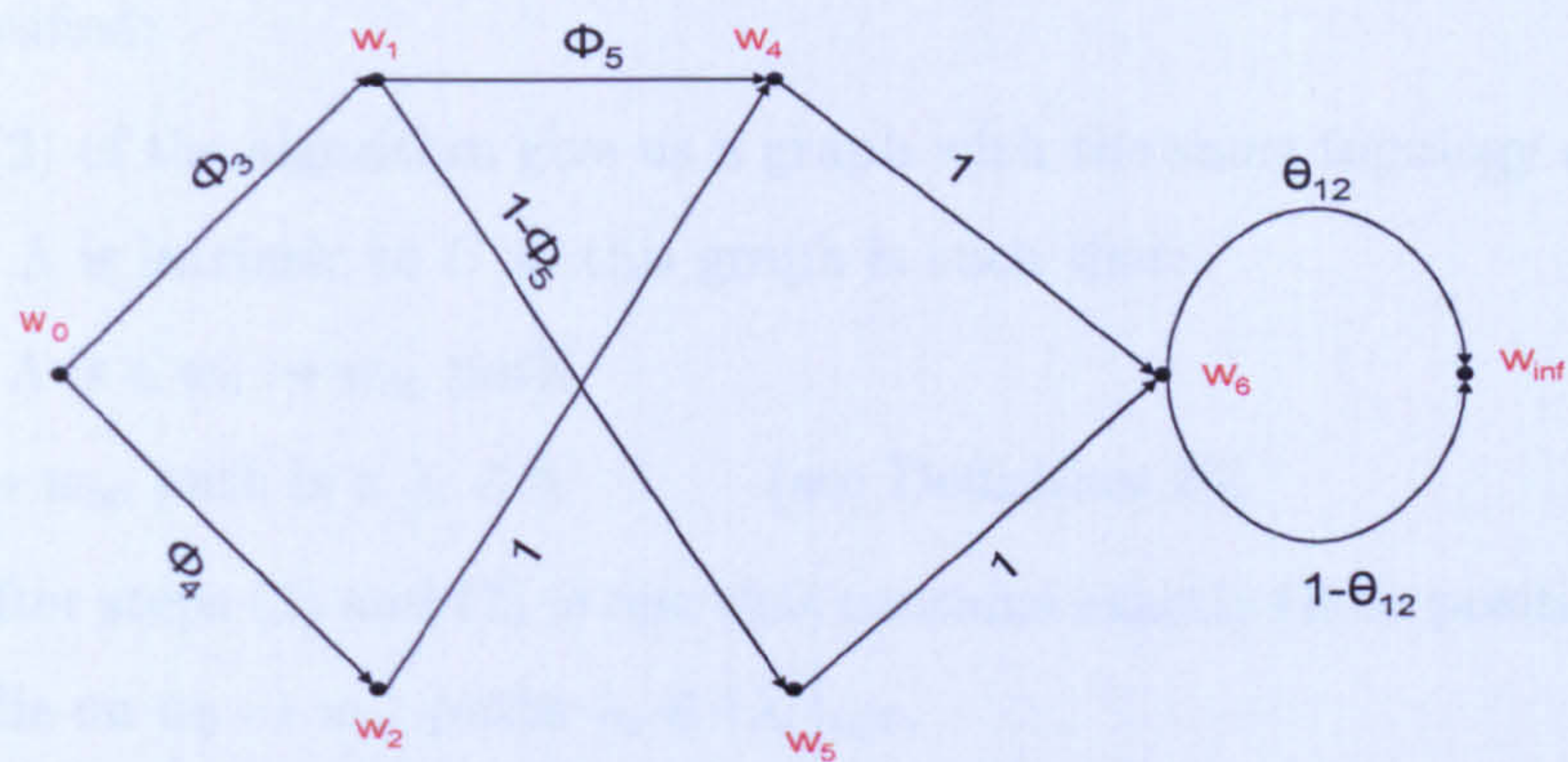


Figure 19: Graph following Algorithm step (3)

Algorithm step (4) (with relabelling of positions) gives us Figure 20; and Figure 21 shows our conditioned CEG C_Λ in the same format as the original CEG in Figure 1:

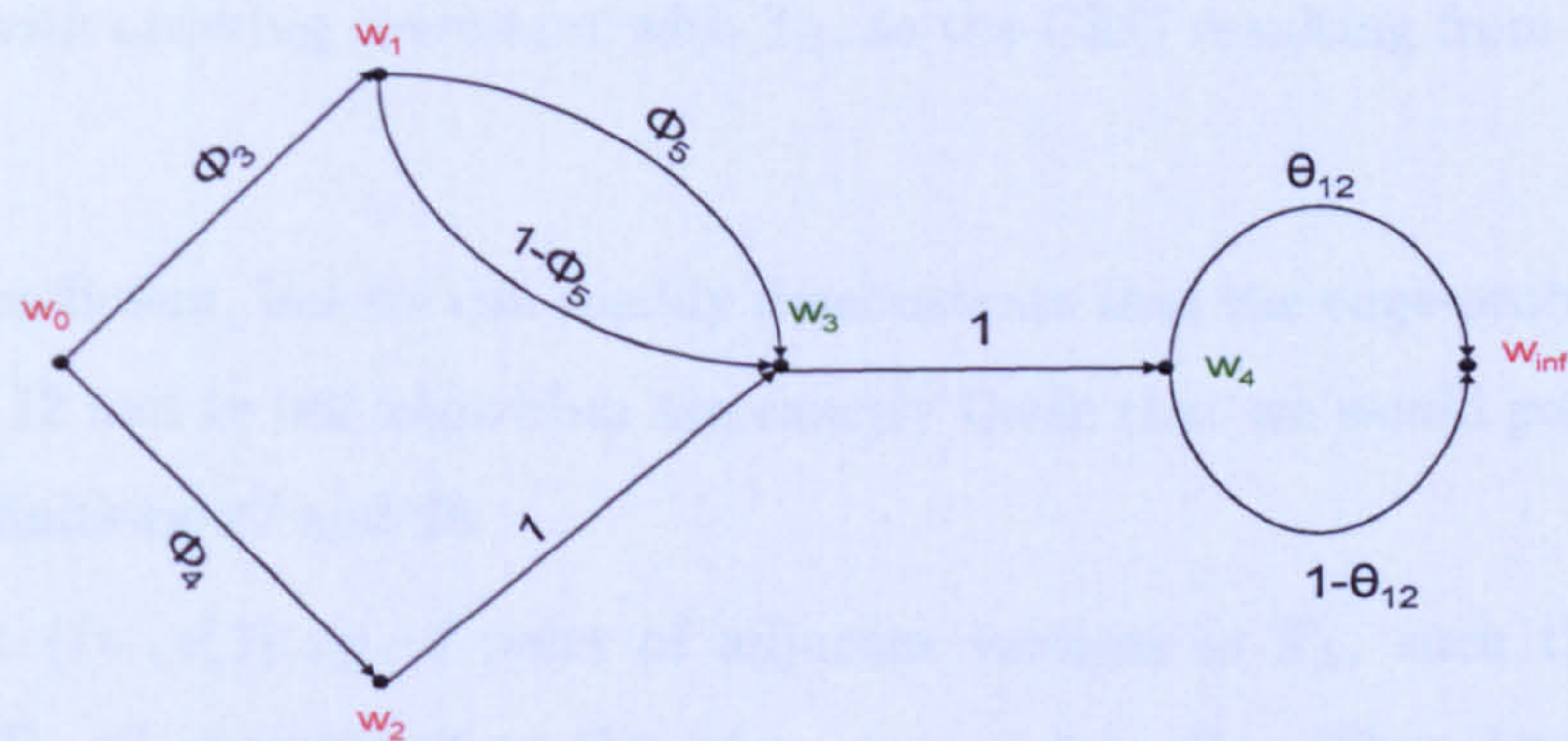


Figure 20: CEG following completion of our algorithm

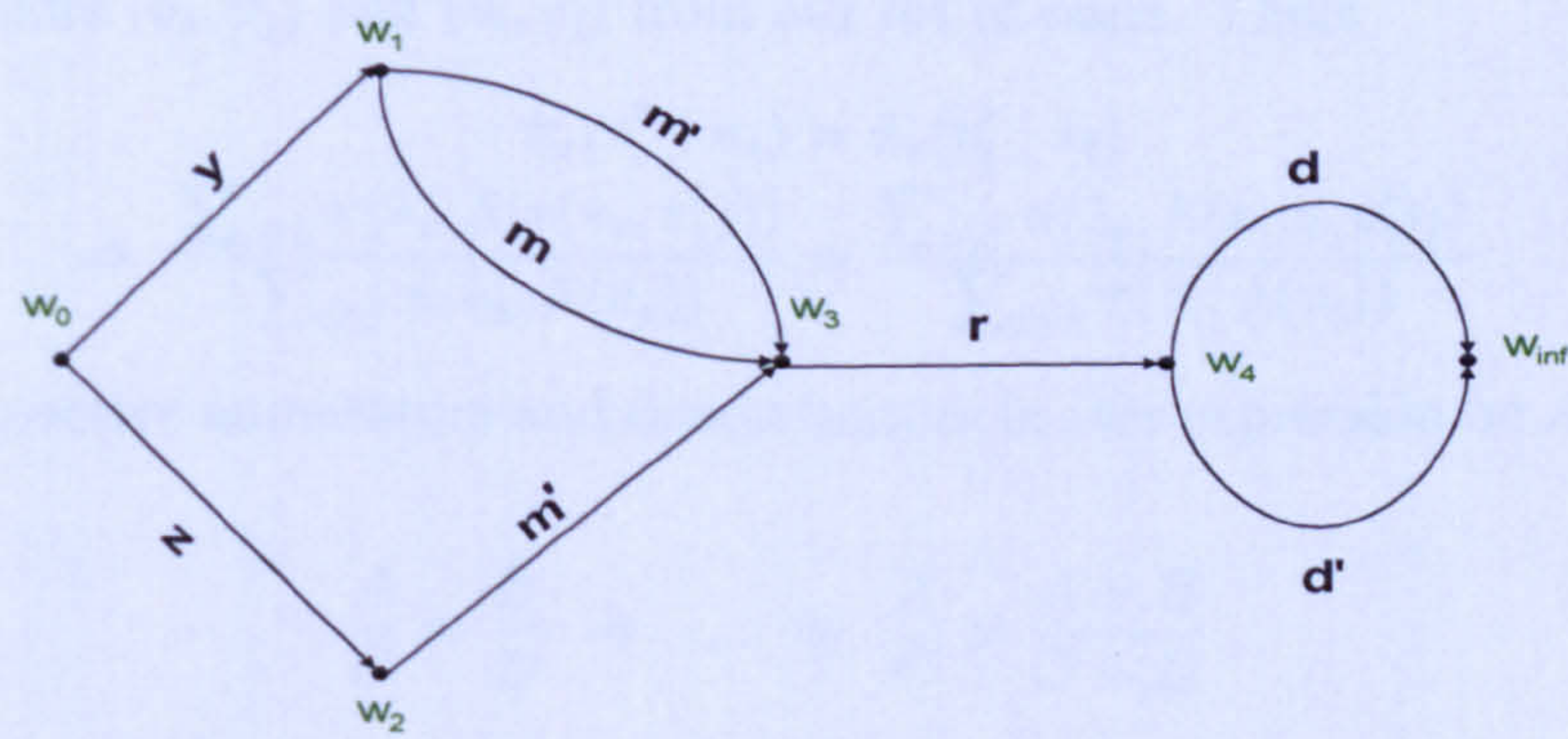


Figure 21: Conditioned CEG C_Λ

We have stated that the graph produced by this new algorithm is C_Λ . This statement needs to be justified:

Steps (1) and (2) of the algorithm give us a graph with the same topology as $G_\Lambda(C)$ (see Definition 25). Λ is intrinsic to C so this graph is such that:

- (i) every $\lambda_i \in \Lambda$ is a $w_0 \rightarrow w_\infty$ path
- (ii) every $w_0 \rightarrow w_\infty$ path is a $\lambda_i \in \Lambda$ (see Definition 26)

So the graph after steps (1) and (2) is one that contains exactly those positions and edges from C which lie on $w_0 \rightarrow w_\infty$ paths $\lambda_i \in \{\lambda_i\}_{i \in I}$.

Step (4) uses the principles of equivalence and stage-equivalence from chapter 2 to combine positions or add undirected edges between positions as appropriate. Hence the graph produced after step (4) is a CEG.

As this graph is derived from C , it has the same ordering as T and hence as T_Λ . From Definition 9 in chapter 2, it is clear that for any Tree T , there exists only one possible CEG $C(T)$ whose ordering is consistent with that of T . Hence there is only one possible CEG $C_\Lambda(T_\Lambda)$ with ordering consistent with T_Λ , so the CEG resulting from our algorithm is C_Λ .

This is in fact sufficient, but we can readily demonstrate that the edge-probabilities given in Proposition 12 and in our algorithm are exactly those that we would get if we started solely with Definitions 27 and 28:

Consider a set $\{(v_r, v'_r)\}_{r \in R}$ of pairs of adjacent vertices in T_Λ , such that the edges $\{e(v_r, v'_r)\}$ in T_Λ all correspond to the edge $e(w_1, w_2)$ in C_Λ . Then by the definition of C_Λ :

$$\hat{\pi}_e(w_2 \mid w_1) = \hat{\pi}_e(v'_r \mid v_r) \quad \forall r \in R$$

Consider two pairs (v_s, v'_s) and (v_t, v'_t) from our set of pairs. Then:

$$\begin{aligned} \hat{\pi}_e(v'_s \mid v_s) &= \hat{\pi}_e(v'_t \mid v_t) \\ \Rightarrow \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(v_s, v'_s)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(v_s))} &= \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(v_t, v'_t)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(v_t))} \end{aligned}$$

Letting the respective numerators and denominators in this expression be A, B, C and D , we get:

$$\frac{A}{C} = \frac{B}{D} \Rightarrow \dots \Rightarrow \frac{A}{C} = \frac{A+B}{C+D}$$

Generalising this result we get:

$$\hat{\pi}_e(v'_s \mid v_s) = \frac{\sum_{r \in R} \sum_{i \in I} \pi(\lambda_i, \Lambda(e(v_r, v'_r)))}{\sum_{r \in R} \sum_{i \in I} \pi(\lambda_i, \Lambda(v_r))}$$

Now, for fixed i, r :

$$\lambda_i \cap \Lambda(v_r) = \begin{cases} \lambda_i & \text{if } \lambda_i \text{ passes through } v_r \\ \phi & \text{otherwise} \end{cases}$$

So:

$$\pi(\lambda_i \cap \Lambda(v_r)) = \begin{cases} \pi(\lambda_i) & \text{if } \lambda_i \text{ passes through } v_r \\ 0 & \text{otherwise} \end{cases}$$

By the relationship between the set $\{v_r\}$ and the position w_1 , λ_i can pass through no more than one v_r for $r \in R$ (Note: This relies on the assumption that two vertices in a Tree lying on the same path cannot be equivalent – see chapter 2). So for fixed i :

$$\sum_{r \in R} \pi(\lambda_i \cap \Lambda(v_r)) = \begin{cases} \pi(\lambda_i) & \text{if } \lambda_i \text{ passes through one of the } v_r \text{ (} r \in R \text{)} \\ 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \sum_{r \in R} \pi(\lambda_i \cap \Lambda(v_r)) = \pi(\lambda_i, \bigcup_{r \in R} \Lambda(v_r))$$

and similarly for $\lambda_i \cap \Lambda(e(v_r, v'_r))$.

So:

$$\begin{aligned} \hat{\pi}_e(v'_s \mid v_s) &= \frac{\sum_{i \in I} \pi(\lambda_i, \bigcup_{r \in R} \Lambda(e(v_r, v'_r)))}{\sum_{i \in I} \pi(\lambda_i, \bigcup_{r \in R} \Lambda(v_r))} \\ \Rightarrow \hat{\pi}_e(w_2 \mid w_1) &= \frac{\sum_{i \in I} \pi(\lambda_i, \bigcup_{r \in R} \Lambda(e(v_r, v'_r)))}{\sum_{i \in I} \pi(\lambda_i, \bigcup_{r \in R} \Lambda(v_r))} \end{aligned}$$

But $\Lambda(w_1) = \bigcup_{r \in R} \Lambda(v_r)$ (section 3.2) and analogously

$\Lambda(e(w_1, w_2)) = \bigcup_{r \in R} \Lambda(e(v_r, v'_r))$, giving:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))}$$

as required.

4.8 Local Message Passing: General observation Sets

As with our previous algorithm (sections 4.2 and 4.5) we can rewrite our new algorithm so that it is more directly comparable to the LMP algorithms used for propagation on BNs. Interestingly, we cannot easily adapt our original LMP algorithm from section 4.5 for use with general observation sets, and the reason for this is as follows:

We have noted that if Λ is of the form $\Lambda = \bigcup_{w_a \in W_X} \Lambda(w_a)$, then we can write $\Phi(w) = \pi(\Lambda \mid w)$ for any $w \prec W_X$. We can do this because for $w_1 \prec w_2 \prec w_a$ and $w_1 \sim w_2$ we have:

$$\pi(\Lambda(w_a) \mid \Lambda(e(w_1, w_2))) = \pi(\Lambda(w_a) \mid \Lambda(w_1), \Lambda(e(w_1, w_2)))$$

since $\Lambda(e(w_1, w_2)) \subset \Lambda(w_1)$, so $\Lambda(w_1) \cap \Lambda(e(w_1, w_2)) = \Lambda(e(w_1, w_2))$

$$= \pi(\Lambda(w_a) \mid \Lambda(w_2))$$

(see proof of Corollary 2 in section 4.2)

So for $\Lambda = \bigcup_{w_a \in W_X} \Lambda(w_a)$, expression (A) from Proposition 12 in section 4.7 reads:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\pi(\Lambda \mid \Lambda(w_2))}{\pi(\Lambda \mid \Lambda(w_1))} \pi_e(w_2 \mid w_1)$$

(which is expression (1) from Corollary 3 of section 4.2). But if $\Lambda \neq \Lambda(w_a)$ for some $w_1 \prec w_2 \prec w_a$ (or a union of such events), we **cannot** in general write:

$$\pi(\Lambda \mid \Lambda(e(w_1, w_2))) = \pi(\Lambda \mid \Lambda(w_2))$$

and **cannot** consequently simplify expression (A) from Proposition 12 in section 4.7.

Example 4:

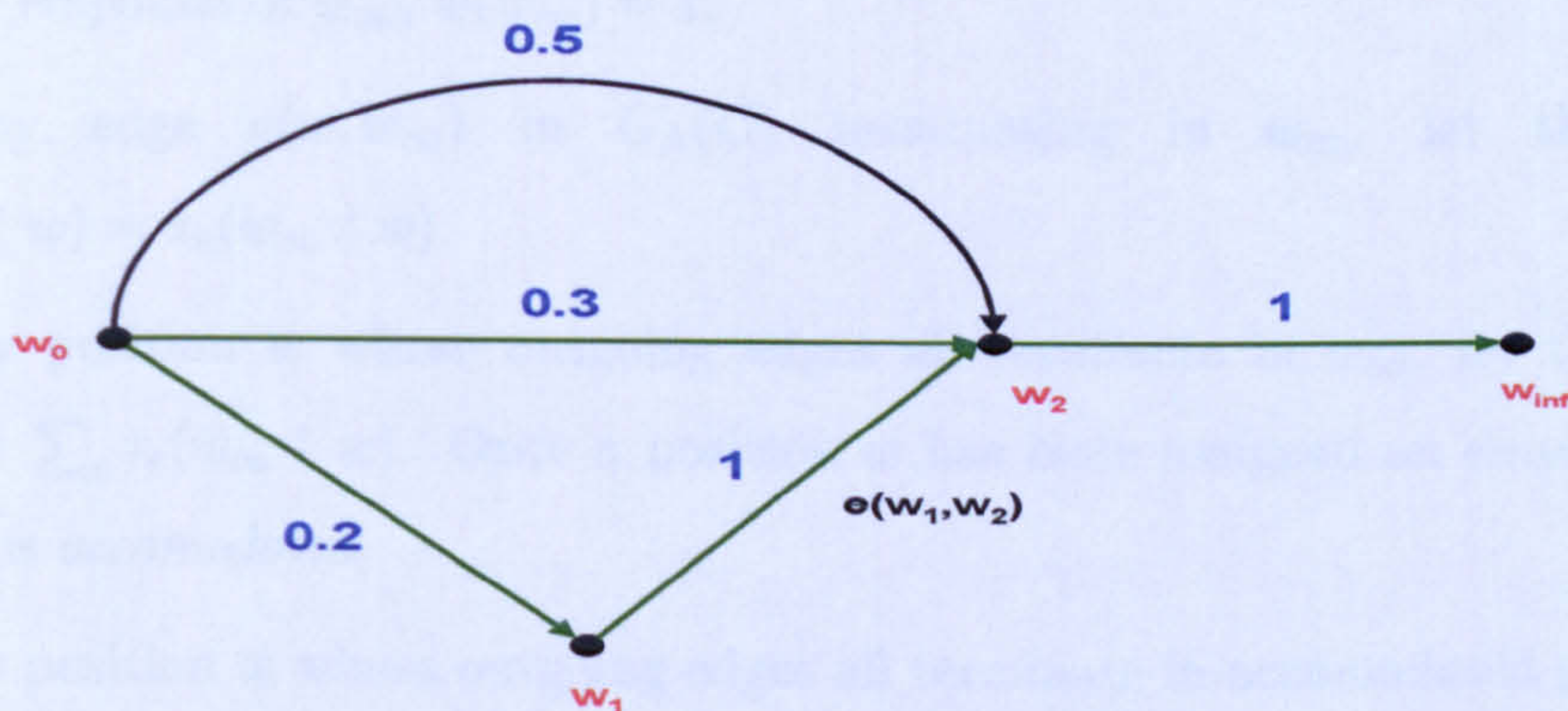


Figure 22: CEG for Example 4

Consider the values of $\pi(\Lambda \mid \Lambda(e(w_1, w_2)))$ and $\pi(\Lambda \mid \Lambda(w_2))$ for the CEG in Figure 22, where Λ is the event outlined in green:

$$\begin{aligned} \pi(\Lambda \mid \Lambda(e(w_1, w_2))) &= \frac{\pi(\Lambda, \Lambda(e(w_1, w_2)))}{\pi(\Lambda(e(w_1, w_2)))} \\ &= \frac{0.2}{0.2} = 1 \end{aligned}$$

whereas

$$\begin{aligned} \pi(\Lambda \mid \Lambda(w_2)) &= \frac{\pi(\Lambda, \Lambda(w_2))}{\pi(\Lambda(w_2))} \\ &= \frac{0.3 + 0.2}{1} = 0.5 \end{aligned}$$

In fact, as we will see, we can use an almost identical process for general observation sets, but the justification of the method is very subtle and requires us to consider the underlying Tree T of our CEG C .

So, consider a CEG C , and an event Λ intrinsic to C . Let T be the Probability Tree associated with C (see Definition 9), T_Λ the Probability Tree associated with C_Λ , and $T_{(\Lambda)}$ the subtree of T containing only those root-to-leaf paths in Λ . $T_{(\Lambda)}$ differs from T_Λ in that the former retains the edge-probabilities from T .

Recall that $G_\Lambda(C)$ is C with all undirected edges removed, and all positions and edges not lying on a $w_0 \rightarrow w_\infty$ path in Λ removed (Definition 25 from section 4.6). As in section 4.5, positions from $W(C)$ are here labelled simply as w , and positions downstream from but adjacent to a specified w are labelled w' .

The Algorithm:

(1) Produce $G_\Lambda(C)$.

Backward step:

- (2) Let the *emphasis* of w_∞ , $\Phi(w_\infty) = 1$.
- (3) For any edge $e(w, w_\infty)$ in $G_\Lambda(C)$ terminating in w_∞ , let the *potential* $\tau_e(w_\infty | w) = \pi_e(w_\infty | w)$.
- (4) For any position w whose outgoing edges all terminate in w_∞ , let the *emphasis* $\Phi(w) = \sum_e \tau_e(w_\infty | w)$. Once a position w has been assigned an emphasis we say that w is *accomodated*.
- (5) For any position w whose outgoing edges all terminate in accomodated positions, let $\tau_e(w' | w) = \pi_e(w' | w) \Phi(w')$.
- (6) For any position w whose outgoing edges all terminate in accomodated positions, let $\Phi(w) = \sum_e \tau_e(w' | w)$.
- (7) Repeat steps (5), (6) until all positions are accomodated.

Forward step:

(8) For any edge $e(w, w')$ in $G_\Lambda(C)$, replace $\tau_e(w' | w)$ by:

$$\hat{\pi}_e(w' | w) = \frac{\tau_e(w' | w)}{\Phi(w)}$$

(9) In the graph produced by steps (1) to (8), any positions which are now equivalent are combined into a single position. Any positions which are now stage-equivalent are connected by an undirected edge, and edges are coloured as appropriate (see Definition 13).

Note that as we move forward through $G_\Lambda(C)$ (step (8)) the updated probabilities of

$w_0 \rightarrow w$ subpaths will be of the form:

$$\hat{\pi}_\mu(w \mid w_0) = \prod_{i=0} \hat{\pi}_e(w_{i+1} \mid w_i)$$

and we get:

$$\hat{\pi}(\Lambda(w)) = \sum_{\mu \in \{\mu(w_0, w)\}} \hat{\pi}_\mu(w \mid w_0)$$

A version of this algorithm also appears in [63], which also includes a pseudo-code version of the algorithm.

We claim that $\hat{\pi}_e(w' \mid w)$ as assigned in step (8) is the correct updated edge-probability; that is:

$$\hat{\pi}_e(w' \mid w) = \frac{\pi(\Lambda \mid \Lambda(e(w, w')))}{\pi(\Lambda \mid \Lambda(w))} \pi_e(w' \mid w)$$

from Proposition 12 in section 4.7.

Proof:

Consider a position $w \in C$ ($w \in G_\Lambda(C)$) corresponding to a set of situations/vertices $\{v_i\} \in T$. Then the subtrees rooted in each v_i are identical both in topology and in their edge-probabilities.

If there is a subpath $\mu(w_0, w)$ which is not part of a $w_0 \rightarrow w_\infty$ path in Λ (ie. $\mu(w_0, w)$ exists in C , but not in $G_\Lambda(C)$) then there will exist a subset of $\{v_i\}$ which does not exist in T_Λ (or $T_{(\Lambda)}$). We split the set $\{v_i\}$ into:

$$\begin{array}{ll} \{v_i\}_{i \in I} & \text{vertices existing in } T_\Lambda \\ \{v_i\}_{i \in J} & \text{vertices not existing in } T_\Lambda \end{array}$$

Because Λ is intrinsic to C , the subtrees in $T_{(\Lambda)}$ rooted in each $v_i \in \{v_i\}_{i \in I}$ are also identical both in topology and in their edge-probabilities that they retain from T .

Suppose there exists an edge $e(w, w')$ in C , then for each $v_i \in \{v_i\}$, there exists an edge $e(v_i, v'_i)$ in T corresponding to this edge. Note that:

$$\Lambda(w) = \bigcup_{i \in I \cup J} \Lambda(v_i)$$

$$\Lambda(e(w, w')) = \bigcup_{i \in I \cup J} \Lambda(e(v_i, v'_i))$$

Note also that:

$$\pi_e(v'_i \mid v_i) = \pi_e(w' \mid w) \quad \forall i \in I \cup J$$

and since the subtrees in $T_{(\Lambda)}$ rooted in each $v_i \in \{v_i\}_{i \in I}$ are identical, we also have:

$$\pi(\Lambda \mid \Lambda(v_i)) = \pi(\Lambda \mid \Lambda(v_j)) \quad \text{for } i, j \in I$$

$[\pi(\Lambda \mid \Lambda(v_i))]$ is the sum of the probabilities of all the $\mu(v_i, v_{leaf})$ subpaths in $T_{(\Lambda)}$

Clearly therefore also:

$$\pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) = \pi(\Lambda, \Lambda(e(v_j, v'_j)) \mid \Lambda(v_j)) \quad \text{for } i, j \in I$$

$[\pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i))]$ is the sum of the probabilities of all the $\mu(v_i, e(v_i, v'_i), v'_i, v_{leaf})$ subpaths in $T_{(\Lambda)}$

So:

$$\frac{\pi(\Lambda \mid \Lambda(e(w, w')))}{\pi(\Lambda \mid \Lambda(w))} \pi_e(w' \mid w) = \frac{\pi(\Lambda \mid \Lambda(w), \Lambda(e(w, w'))) \pi(\Lambda(e(w, w')) \mid \Lambda(w))}{\pi(\Lambda \mid \Lambda(w))}$$

since $\Lambda(w) \supset \Lambda(e(w, w'))$ in C

$$\begin{aligned} &= \frac{\pi(\Lambda, \Lambda(w), \Lambda(e(w, w')))}{\pi(\Lambda, \Lambda(w))} \\ &= \frac{\pi(\Lambda, \bigcup_{i \in I \cup J} [\Lambda(v_i), \Lambda(e(v_i, v'_i))])}{\pi(\Lambda, \bigcup_{i \in I \cup J} \Lambda(v_i))} \end{aligned}$$

(an expression evaluated on T) since $\Lambda(v_i) \cap \Lambda(e(v_j, v'_j)) = \emptyset$ for $i \neq j$

$$= \frac{\sum_{i \in I \cup J} \pi(\Lambda, \Lambda(v_i), \Lambda(e(v_i, v'_i)))}{\sum_{i \in I \cup J} \pi(\Lambda, \Lambda(v_i))}$$

But $\Lambda \cap \Lambda(v_i) = \emptyset$ for $v_i \in \{v_i\}_{i \in J}$, so this equals:

$$\begin{aligned} &= \frac{\sum_{i \in I} \pi(\Lambda, \Lambda(v_i), \Lambda(e(v_i, v'_i)))}{\sum_{i \in I} \pi(\Lambda, \Lambda(v_i))} \\ &= \frac{\sum_{i \in I} \pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) \pi(\Lambda(v_i))}{\sum_{i \in I} \pi(\Lambda \mid \Lambda(v_i)) \pi(\Lambda(v_i))} \\ &= \frac{\pi(\Lambda, \Lambda(e(v_j, v'_j)) \mid \Lambda(v_j)) \sum_{i \in I} \pi(\Lambda(v_i))}{\pi(\Lambda \mid \Lambda(v_j)) \sum_{i \in I} \pi(\Lambda(v_i))} \end{aligned}$$

for any $v_j \in \{v_i\}_{i \in I}$

$$= \frac{\pi(\Lambda, \Lambda(e(v_j, v'_j)) \mid \Lambda(v_j))}{\pi(\Lambda \mid \Lambda(v_j))}$$

for any $v_j \in \{v_i\}_{i \in I}$

We now turn our attention to the terms in the algorithm:

Consider positions $w \in G_{\Lambda}(C)$, all of whose outgoing edges terminate in w_{∞} . Then:

$$\begin{aligned} \Phi(w) &= \sum_e \tau_e(w_{\infty} \mid w) \\ &= \sum_e \pi_e(w_{\infty} \mid w) \quad \text{in } G_{\Lambda}(C) \end{aligned}$$

Now each edge $e(w, w_\infty)$ in $G_\Lambda(C)$ corresponds to an edge $e(v_i, v_{leaf})$ in $T_{(\Lambda)}$, and has the same probability as this edge. So this sum is equal to the sum of the probabilities of all the $e(v_i, v_{leaf})$ edges in $T_{(\Lambda)}$. That is:

$$\Phi(w) = \pi(\Lambda \mid \Lambda(v_i))$$

for any $v_i \in \{v_i\}_{i \in I}$

We claim that $\Phi(w) = \pi(\Lambda \mid \Lambda(v_i))$ ($v_i \in \{v_i\}_{i \in I}$) for all $w \in G_\Lambda(C)$, where $\{v_i\}_{i \in I}$ is the set of vertices in $T_{(\Lambda)}$ corresponding to w .

We prove this by induction:

Step 1.

Let v_{leaf} be a vertex in $T_{(\Lambda)}$ (corresponding to w_∞). Then $\Phi(w_\infty) = 1 = \pi(\Lambda \mid \Lambda(v_{leaf}))$. Also, as we have just seen, if w is such that all of its outgoing edges terminate in w_∞ , then $\Phi(w) = \pi(\Lambda \mid \Lambda(v_i))$.

Step 2.

Suppose $w \in G_\Lambda(C)$ has a set of outgoing edges $\{e(w, w')\}$ terminating in a set of positions $\{w'\}$. Each of these edges in turn corresponds to a set of edges $\{e(v_i, v'_i)\}$ in $T_{(\Lambda)}$. So, if we choose a specific v_i corresponding to w , then we will have a specific v'_i corresponding to w' .

But, as with w , w' will correspond to a set of vertices in T some of which exist in T_Λ , and some of which do not. Let the former set be $\{v'_i\}_{i \in I'}$.

Now suppose that for each $w' \in \{w'\}$, we can write $\Phi(w') = \pi(\Lambda \mid \Lambda(v'_i))$, for any $v'_i \in \{v'_i\}_{i \in I'}$. Then if we choose a specific $v_i \in T_{(\Lambda)}$ corresponding to $w \in G_\Lambda(C)$, then our consequent v'_i corresponding to w' will exist in $T_{(\Lambda)}$, and so be a member of $\{v'_i\}_{i \in I'}$. Hence we can write $\Phi(w') = \pi(\Lambda \mid \Lambda(v'_i))$ for this v'_i .

So:

$$\begin{aligned} \Phi(w) &= \sum_e \tau_e(w' \mid w) \\ &= \sum_e \pi_e(w' \mid w) \Phi(w') \end{aligned}$$

As the edge $e(w, w') \in G_\Lambda(C)$ has the same probability as the edge $e(v_i, v'_i)$ in $T_{(\Lambda)}$, and as there is a 1:1 correspondence between edges $e(v_i, v'_i)$ and vertices v'_i (since we are working on a Tree), we can therefore write:

$$\Phi(w) = \sum_{v'_i} \pi_e(v'_i \mid v_i) \pi(\Lambda \mid \Lambda(v'_i)) \quad \text{evaluated on } T_{(\Lambda)}$$

for any $v_i \in \{v_i\}_{i \in I}$ and $v'_i \in \{v'_i \mid e(v_i, v'_i) \text{ exists in } T_\Lambda\}$.

So:

$$\Phi(w) = \sum_{v'_i} \pi(e(v_i, v'_i) \mid \Lambda(v_i)) \pi(\Lambda \mid \Lambda(v'_i))$$

But $\Lambda(v'_i) = \Lambda(e(v_i, v'_i)) \subset \Lambda(v_i)$ in a Tree, so this equals:

$$\begin{aligned} & \sum_{v'_i} \pi(\Lambda(e(v_i, v'_i)), \Lambda(v'_i) \mid \Lambda(v_i)) \pi(\Lambda \mid \Lambda(v_i), \Lambda(e(v_i, v'_i)), \Lambda(v'_i)) \\ &= \sum_{v'_i} \pi(\Lambda, \Lambda(e(v_i, v'_i)), \Lambda(v'_i) \mid \Lambda(v_i)) \\ &= \sum_{v'_i} \pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) \\ &= \pi(\Lambda, \bigcup_{v'_i} \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) \end{aligned}$$

But as this union is over $\{v'_i \mid e(v_i, v'_i) \text{ exists in } T_\Lambda\}$, $\Lambda \cap \left[\bigcup_{v'_i} \Lambda(e(v_i, v'_i)) \right] = \Lambda \cap \Lambda(v_i)$, and hence:

$$\begin{aligned} \Phi(w) &= \pi(\Lambda, \Lambda(v_i) \mid \Lambda(v_i)) \\ &= \pi(\Lambda \mid \Lambda(v_i)) \end{aligned}$$

□

We now turn our attention to the potentials:

$$\begin{aligned} \tau_e(w' \mid w) &= \pi_e(w' \mid w) \Phi(w') \\ &= \pi_e(v'_i \mid v_i) \pi(\Lambda \mid \Lambda(v'_i)) \end{aligned}$$

for any $v_i \in \{v_i\}_{i \in I}$ and $v'_i \in \{v'_i \mid e(v_i, v'_i) \text{ exists in } T_\Lambda\}$

$$= \pi(\Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) \pi(\Lambda \mid \Lambda(v'_i))$$

But $\Lambda(v'_i) = \Lambda(e(v_i, v'_i)) \subset \Lambda(v_i)$ in a Tree, so this equals:

$$\begin{aligned} & \pi(\Lambda(e(v_i, v'_i)), \Lambda(v'_i) \mid \Lambda(v_i)) \pi(\Lambda \mid \Lambda(v_i), \Lambda(e(v_i, v'_i)), \Lambda(v'_i)) \\ &= \pi(\Lambda, \Lambda(e(v_i, v'_i)), \Lambda(v'_i) \mid \Lambda(v_i)) \\ &= \pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i)) \end{aligned}$$

Substituting our new expressions for the emphasis $\Phi(w)$ and the potential $\tau_e(w' \mid w)$ into the expression in step (8) of the algorithm, we get:

$$\begin{aligned} \hat{\pi}_e(w' \mid w) &= \frac{\tau_e(w' \mid w)}{\Phi(w)} \\ &= \frac{\pi(\Lambda, \Lambda(e(v_i, v'_i)) \mid \Lambda(v_i))}{\pi(\Lambda \mid \Lambda(v_i))} \\ &= \frac{\pi(\Lambda \mid \Lambda(e(w, w')))}{\pi(\Lambda \mid \Lambda(w))} \pi_e(w' \mid w) \end{aligned}$$

□

We conclude this section with a context-free example illustrating our new LMP algorithm. A slight variation of this example appears in [63].

Example 5:

Consider the CEG in Figure 23:

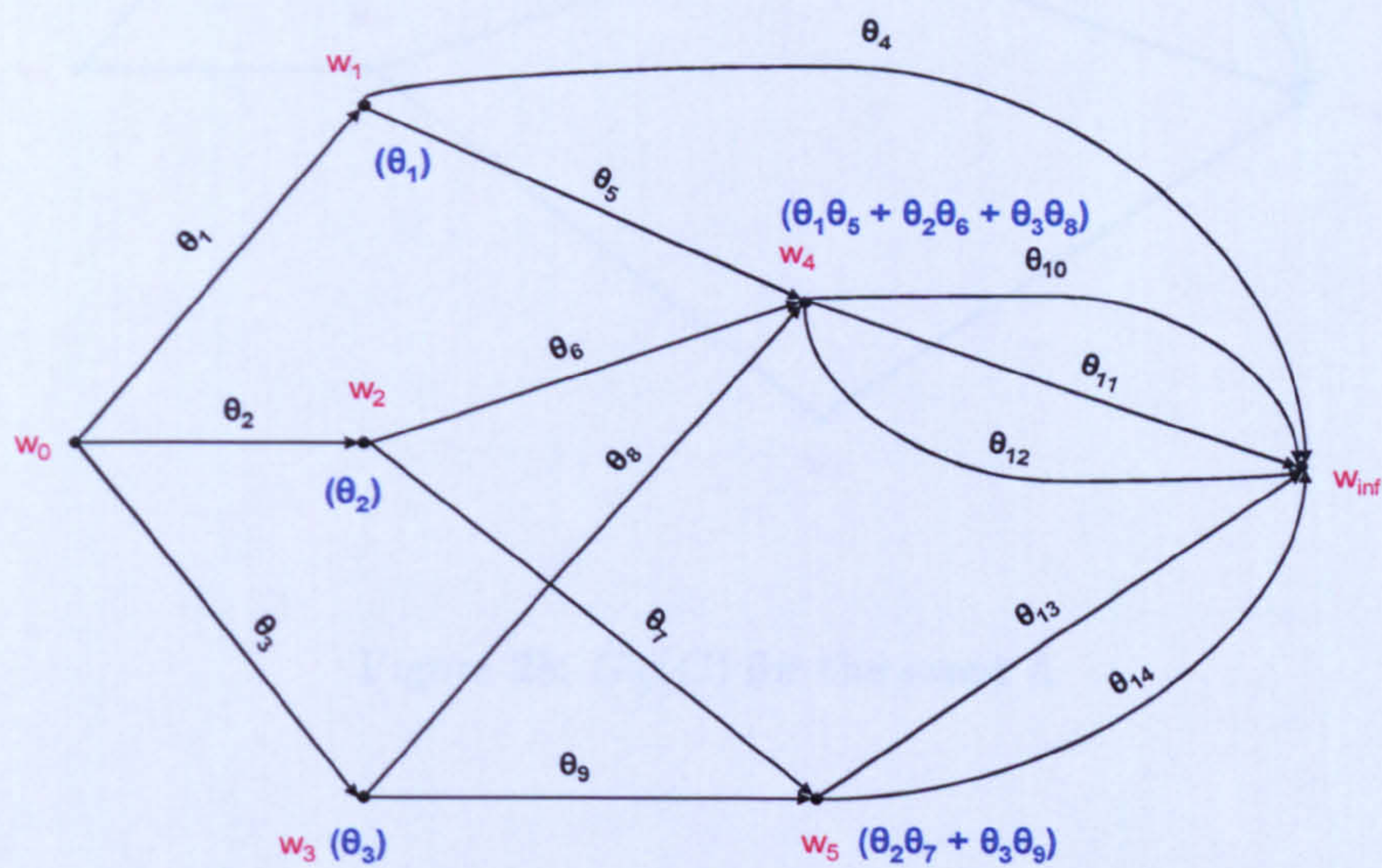


Figure 23: CEG C for our example

Note that each edge has been labelled with a probability $\pi_e(w' \mid w)$, and each position with a probability $\pi(\Lambda(w))$. We consider the event Λ outlined in green in Figure 24:

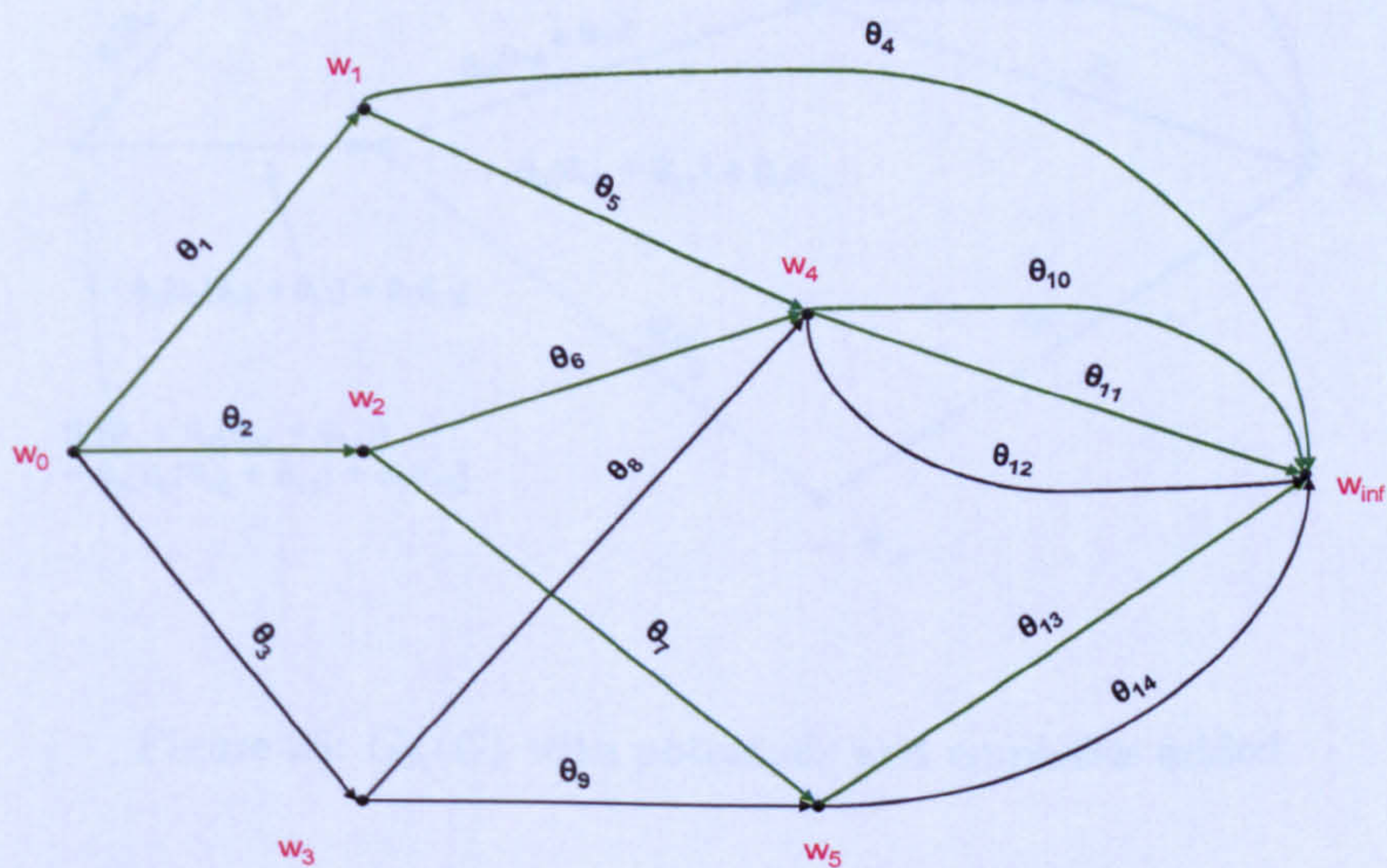


Figure 24: CEG C with event Λ indicated

Step (1) of the LMP algorithm gives us $G_\Lambda(C)$, shown in Figure 25:

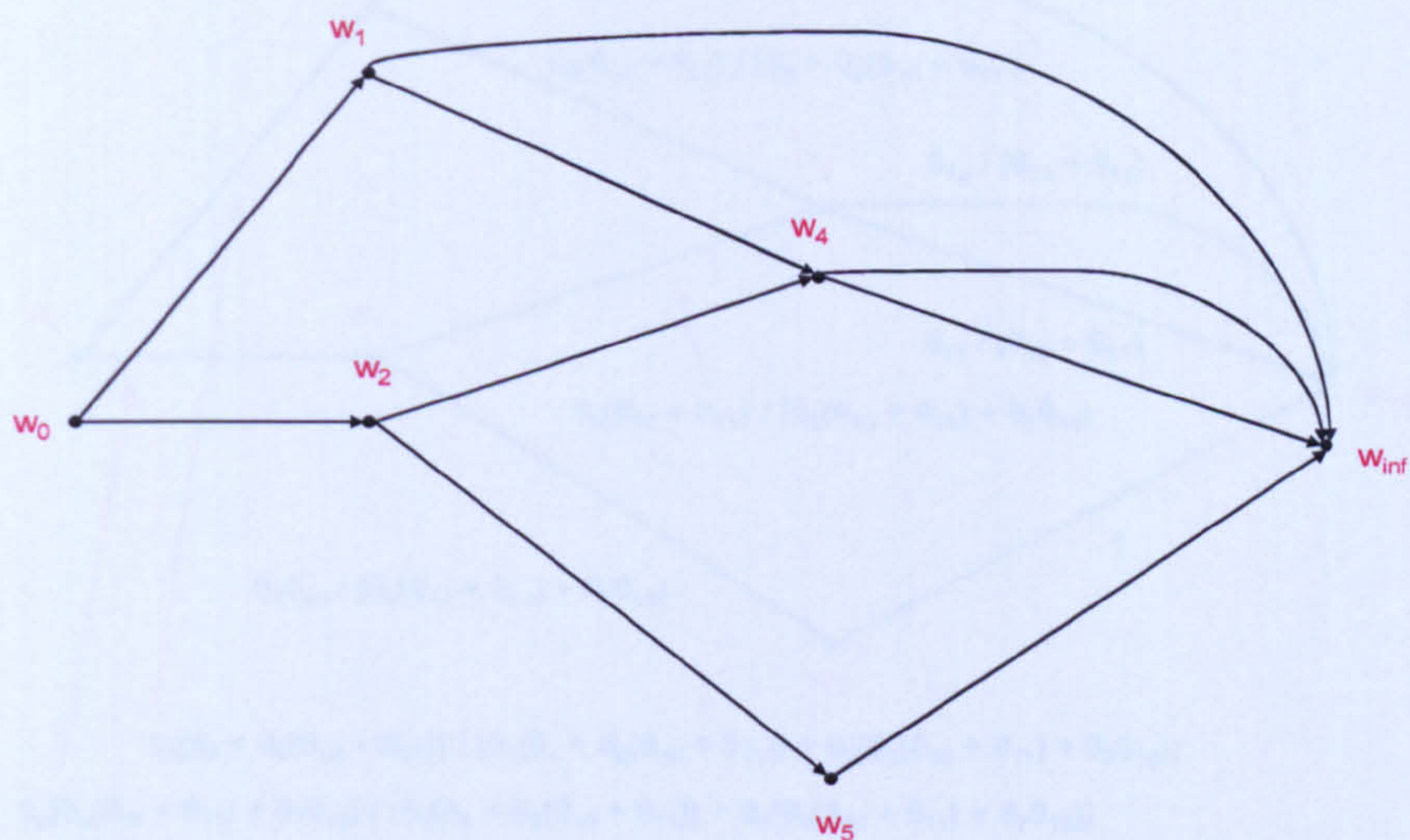


Figure 25: $G_\Lambda(C)$ for the event Λ

Steps (2) through (7) give us the graph in Figure 26, in which the potentials $\tau_e(w' | w)$ and emphases $\Phi(w)$ have been added to the edges and positions of $G_\Lambda(C)$:

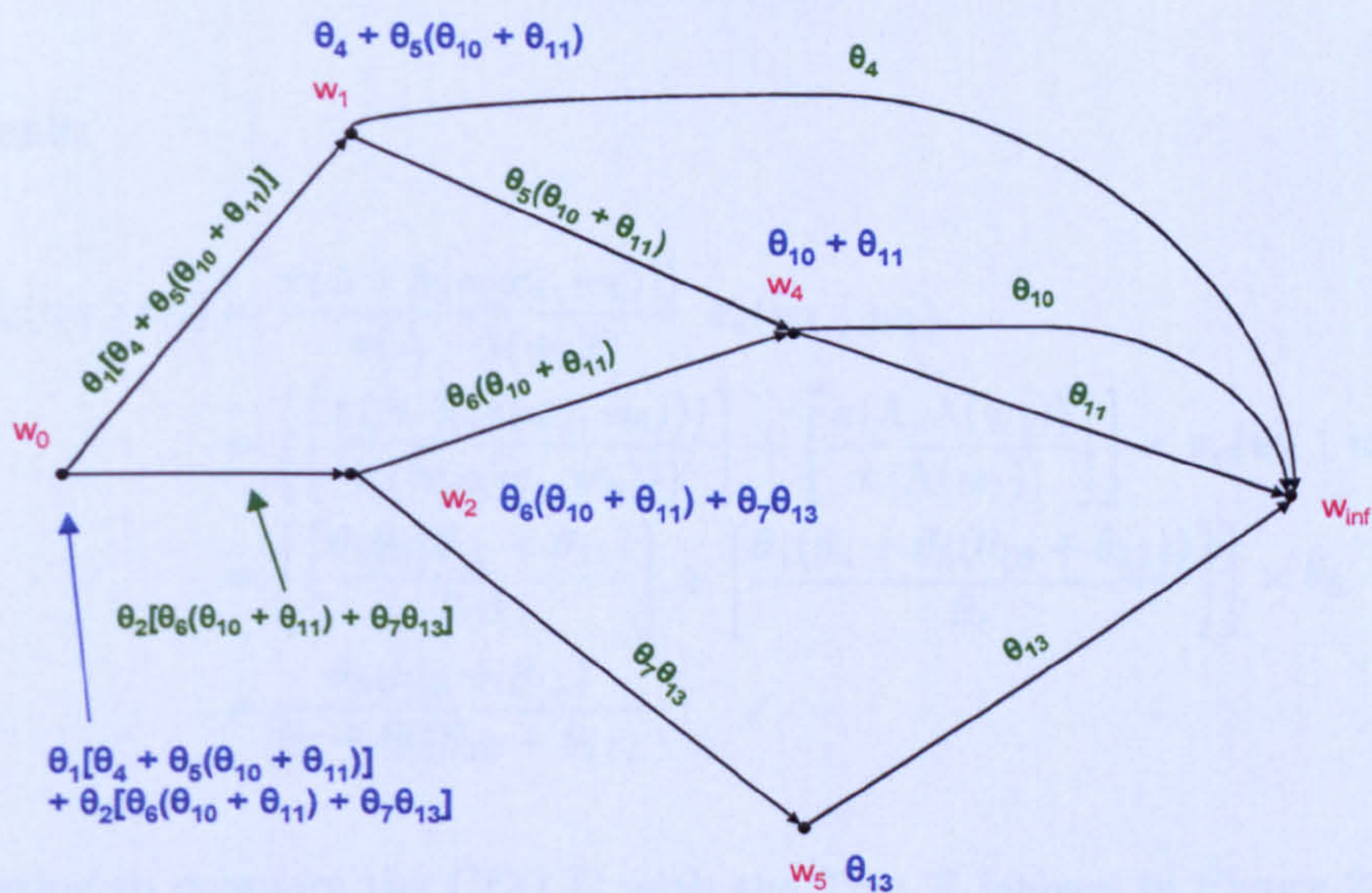


Figure 26: $G_\Lambda(C)$ with potentials and emphases added

Step (8) puts updated probabilities $\hat{\pi}_e(w' | w)$ onto the edges of $G_\Lambda(C)$ to give us the updated CEG shown in Figure 27:

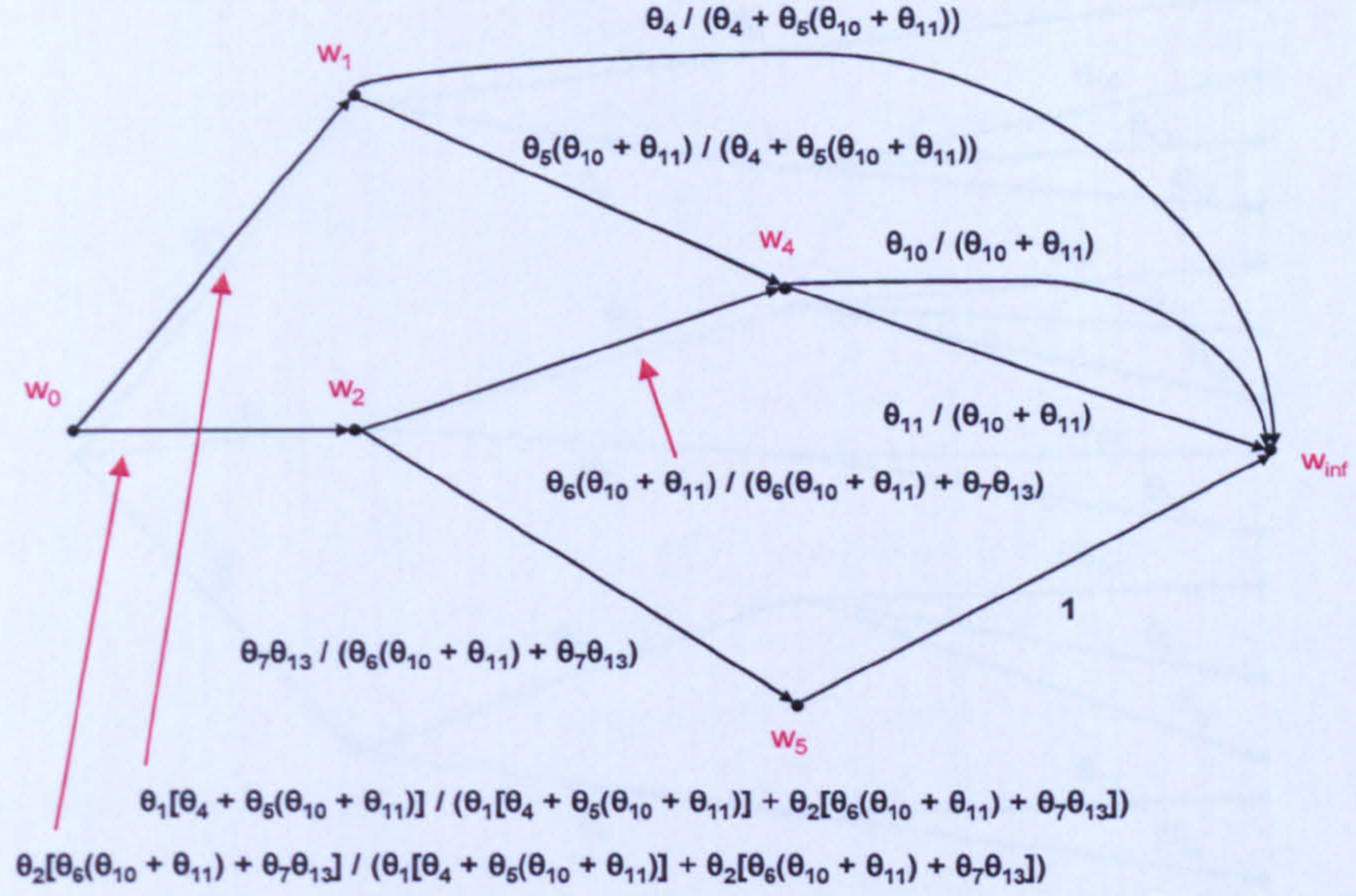


Figure 27: Updated CEG (with unchanged position labels)

We can check these updated edge-probabilities using the expression:

$$\hat{\pi}_e(w' | w) = \frac{\pi(\Lambda | \Lambda(e(w, w')))}{\pi(\Lambda | \Lambda(w))} \pi_e(w' | w)$$

so for example:

$$\begin{aligned} \hat{\pi}_e(w_4 | w_1) &= \frac{\pi(\Lambda | \Lambda(e(w_1, w_4)))}{\pi(\Lambda | \Lambda(w_1))} \pi_e(w_4 | w_1) \\ &= \left[\frac{\pi(\Lambda, \Lambda(e(w_1, w_4)))}{\pi(\Lambda(e(w_1, w_4)))} \right] \div \left[\frac{\pi(\Lambda, \Lambda(w_1))}{\pi(\Lambda(w_1))} \right] \times \pi_e(w_4 | w_1) \\ &= \left[\frac{\theta_1\theta_5(\theta_{10} + \theta_{11})}{\theta_1\theta_5} \right] \div \left[\frac{\theta_1(\theta_4 + \theta_5(\theta_{10} + \theta_{11}))}{\theta_1} \right] \times \theta_5 \\ &= \frac{\theta_5(\theta_{10} + \theta_{11})}{\theta_4 + \theta_5(\theta_{10} + \theta_{11})} \quad \checkmark \end{aligned}$$

It is interesting to compare the CEG C with the Tree T (shown in Figure 28, with $T_{(\Lambda)}$ outlined in green). Note that:

w_4 corresponds to $\{v_4^i\}$ and that v_4^1, v_4^2 exist in T_Λ , but that v_4^3 does not.

Notice that the subtrees rooted in v_4^1 and v_4^2 in $T_{(\Lambda)}$ are identical.

Note also that:

w_5 corresponds to $\{v_5^i\}$ and that v_5^1 exists in T_Λ , but that v_5^2 does not.

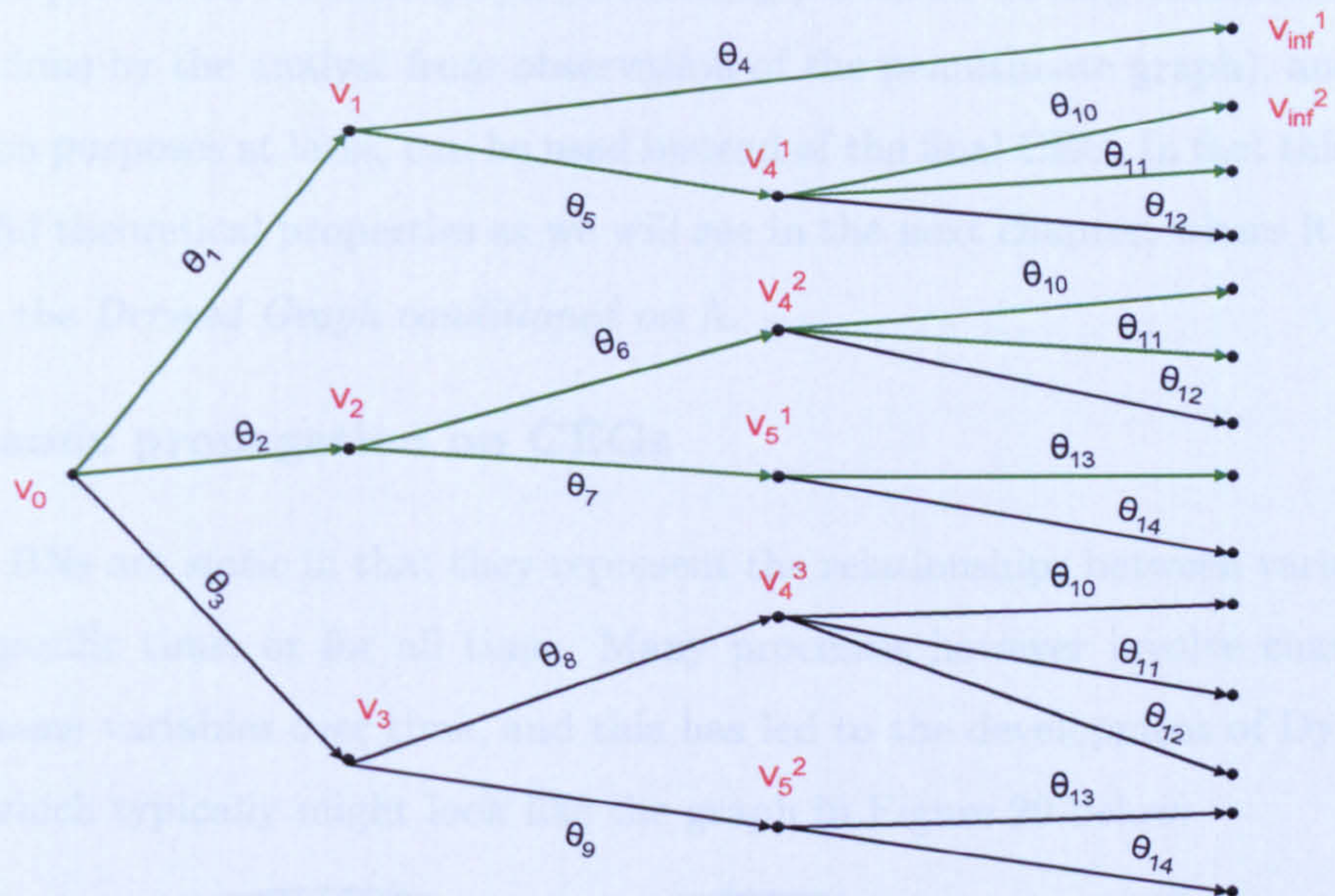


Figure 28: Tree T with $T_{(\Lambda)}$ outlined in green

We see that:

$$\begin{aligned}\Phi(w_4) &= \theta_{10} + \theta_{11} = \pi(\Lambda \mid v_4^1) = \pi(\Lambda \mid v_4^2) \\ \pi(\Lambda \mid v_4^3) &= 0 \neq \Phi(w_4) \\ \pi(\Lambda \mid w_4) &= \frac{(\theta_1\theta_5 + \theta_2\theta_6)(\theta_{10} + \theta_{11})}{\theta_1\theta_5 + \theta_2\theta_6 + \theta_3\theta_8} \neq \Phi(w_4)\end{aligned}$$

Similarly:

$$\begin{aligned}\Phi(w_5) &= \theta_{13} = \pi(\Lambda \mid v_5^1) \\ \pi(\Lambda \mid v_5^2) &= 0 \neq \Phi(w_5) \\ \pi(\Lambda \mid w_5) &= \frac{\theta_2\theta_7\theta_{13}}{\theta_2\theta_7 + \theta_3\theta_9} \neq \Phi(w_5)\end{aligned}$$

So, for $w \in C$ corresponding to $\{v_i\} \in T$, $\Phi(w) = \pi(\Lambda \mid v_i)$ for $v_i \in \{v_i\}_{i \in I}$, vertices existing in T_Λ ($\pi(\Lambda \mid v_i) = 0$ for $v_i \in \{v_i\}_{i \in J}$, vertices not existing in T_Λ). But unlike for events of the type described in section 4.5, $\Phi(w)$ is **not** in general equal to $\pi(\Lambda \mid w)$.

It has been argued (for example by Finn Jensen, in discussion with Jim Smith), that from a computing point of view, changing the topology of a graph is more expensive than changing probabilities on the graph's topology. Now, the pruning of vertices and edges in the early steps of our algorithms is not a major difficulty, but the combining of positions in the final step of our algorithms will be computationally expensive.

As it happens, this is not really a problem, as the updated graph prior to this final step, although not usually a CEG (as I have defined one), none-the-less contains all the

information present in the final graph (combining positions, adding undirected edges and colours is done by the analyst from observation of the penultimate graph), and hence for propagation purposes at least, can be used instead of the final CEG. In fact this graph has many useful theoretical properties as we will see in the next chapter, where it is formally defined as the *Derived Graph conditioned on Λ* .

4.9 Dynamic propagation on CEGs

Typically, BNs are *static* in that they represent the relationships between variables either at some specific time, or for all time. Many processes however involve changes in the values of some variables over time, and this has led to the development of Dynamic BNs (DBNs), which typically might look like the graph in Figure 29 below:

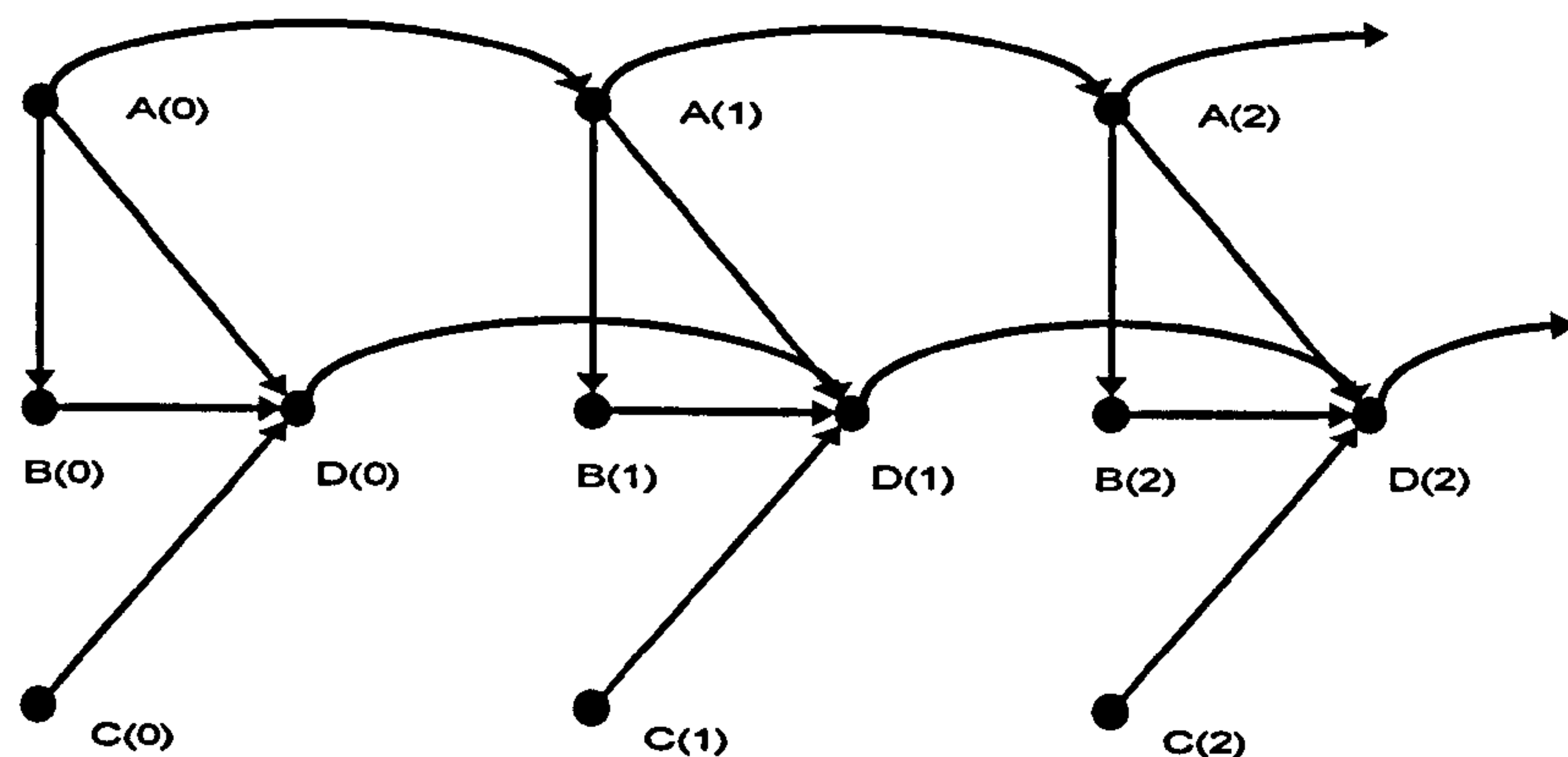


Figure 29: A Dynamic Bayesian Network

This DBN consists of a sequence of *time slices* – *static* BNs connected by *temporal links* [26]. The static BNs each obey the same set of conditional independence statements (ie. they have the same topology). Note that the edges joining the time slices only connect together a subset of the vertices in any time slice.

In principle, one could draw the complete DBN, but for obvious reasons analysis tends to concentrate on a *window* of two or more time slices, from which previous time slices have been pruned. We could visualise this as taking a sequence of camera snapshots along the length of the DBN. For this reason these graphs are often called Dynamic time-sliced BNs.

Note that any information present in the pruned part of the DBN that is needed for current or future analysis must exist somehow in the current window. Also, DBNs need to obey the Markov property that the future is independent of the past given the present. We can see this in Figure 29, as the time slice consisting of the static BN

$\{A(1), B(1), C(1), D(1)\}$ separates the static BN $\{A(0), B(0), C(0), D(0)\}$ from the static BNs $\{A(i), B(i), C(i), D(i)\}_{i \geq 2}$.

Propagation on DBNs involves the observation of values of a subset of the variables in the static BNs at each time point and updating beliefs about the system. Clearly, an observation of $D(2)$ in Figure 29 for example, will have consequences for our beliefs about the values of $A(2), B(2), C(2)$, but also about the values of variables in time slices 1 and 3. The Junction Tree algorithms described in section 4.1 can be adapted to produce algorithms for propagation on DBNs. A good description of the theory of Dynamic Bayesian Networks is available in [27].

Similar principles can be applied both to create dynamic CEGs, and to propagate information on these graphs. In fact the two processes are effectively one as I show below. The remainder of this section offers an introduction to dynamic CEGs – the development of the theory is a task which Jim Smith and I will be tackling in the near future.

Like the DBN, we can envisage the dynamic CEG as a sequence of connected *static* CEGs (each of whose paths have **not** been constrained to converge to a single sink-vertex), and in principle (as with DBNs) we could draw the complete dynamic CEG. The problems here are more severe than those encountered when attempting to draw a complete DBN: Obviously this latter graph represents all information at all times, but to do this in our dynamic CEG, it must include $w_0 \rightarrow w_\infty$ paths representing all possible developments of the system, which would result in a vast unmanageable graph. Instead (as with DBNs) we consider *time-sliced* dynamic CEGs.

The simplest case to consider is where our observations within each static CEG are of events that can be expressed in the form $\Lambda = \bigcup_{w \in W_X} \Lambda(w)$, which as indicated in section 4.2 are often the type of events one would be analysing if one was working with a BN.

We call our first time slice or static CEG $C[1]$, and let it be such that $W(C[1])$ contains all $w \in W_1$, where our first observation will be of the event $\Lambda_1 = \bigcup_{w \in W_1} \Lambda(w)$. We then observe the first event Λ_1 , and update our static CEG: $C'[1] = C[1] \mid \Lambda_1$.

Note that the learning process here helps us to **decrease** complexity and keep the CEG manageable: $C'[1]$ is always simpler than $C[1]$ since we prune branches stemming from edges with zero probabilities (no such pruning process is possible on a BN).

We now create a CEG $C[2]$ such that $W(C[2])$ contains all $w \in W_2$, where our second observation will be of the event $\Lambda_2 = \bigcup_{w \in W_2} \Lambda(w)$. $C[2]$ is created not by extending $C[1]$,

but by extending $C'[1]$ downstream. We then observe the second event Λ_2 , and update: $C'[2] = C[2] \mid \Lambda_2$.

So the learning process here actually defines our dynamic CEG, as each new time slice is created by conditioning an earlier time slice on our observation. Also, this is dynamic learning as each new observation Λ_i allows us to update our CEG and prune all redundant branches. We can of course prune previous time slices, as all necessary information will be stored in the set of positions W_{i-1} (a consequence of the results produced in chapter 3). As already indicated, the key point here is that learning actually reduces the complexity of the CEG, which is not the case for DBNs, where the triangularisation required as new variables arrive tends to increase average clique size, which decreases the efficiency of the propagation algorithm ([44]).

Clearly, as the CEG is much more flexible than the BN, we do not have to confine ourselves to events that can be expressed as $\Lambda = \bigcup \Lambda(w)$. Fast Junction Tree algorithms for BNs assume that observations need to be of subsets of the variables depicted by the vertices of the BN. This is not the case with a dynamic CEG where our observations can be of any set of paths within our static CEG or time slice. Moreover, there is no need for our static CEGs all to have the same topology – this topology could change with time, and in principle this would cause us no real extra difficulties in analysis. Clearly the CEG is an ideal vehicle for the learning of both symmetric and asymmetric models in dynamic contexts.

5. Causal manipulation of Chain Event Graphs

5.1 Introduction

The study of Causality dates back to Hume who published [20] in 1739 and [21] in 1748, but for practical purposes, the first work on the subject of major statistical significance was by Granger, who defined causality in terms of a type of temporal association in 1969 [17]. Granger-causality is still used, but others since then have developed the analysis of cause and effect and taken it in several different directions (for example [47][46]).

Interest in the subject has flourished since the late 1980s, with much of the impetus coming from practitioners in econometrics and computer science who wished to do more than simply reveal correlations between variables. Much of the development of the theory has happened in parallel with that for graphical models, facilitated by the arrival of the d-separation theorem. A key publication here was Pearl's [40] which contained his formulation of the Back Door and Front Door theorems, as well as his rules of *do* calculus. Biometrika published nine replies to this paper and a rejoinder from Pearl. This paper, together with [57] by Spirtes et al (published two years earlier) and Pearl's [41], set the theory of Causality firmly within that for BNs. I hope however, in this chapter, to suggest that BNs are not the ideal graphical model for the analysis of cause and effect.

Even in the last twenty years, researchers in different fields have approached the subject from many different angles. One useful line of enquiry is to attempt to deduce causes from observed effects. Closely related to this approach is *counterfactual* analysis. Counterfactuals are outcomes that would have been observed had the world developed differently – assertions of the form *if X had been the case, then Y would have happened*, when it is known to be false that X is the case [11]. The use of counterfactual analysis in the study of causality is widespread, but is not without its opponents.

In [41], Pearl discusses a number of informal definitions of causality before focussing on how *cause* relates to control. It is this area on which I concentrate in this chapter, looking at the idea of cause and effect through the analysis of *controlled* models.

So, in this context, what do we mean by saying that *M is a cause of N* ? Consider the statement *smoking causes cancer*, and let A, B be random variables such that A takes values corresponding to the outcomes {subject is a non-smoker (a_0), subject is a smoker (a_1)}; B takes values corresponding to the outcomes {subject does not develop lung cancer (b_0), subject does develop lung cancer (b_1)}.

Then our statement clearly does not mean $\pi(b_1 | a_1) = 1$. But neither, in the sense used by most statisticians (and lawyers!), does it mean

$$\pi(b_1 | a_1) > \pi(b_1 | a_0)$$

The argument put forward by Pearl [41] is that we need to consider an *intervention* (or *manipulation*), such that our subject is **made** to smoke(!) He calls this *doing*, and such an intervention could be written $do A = a_1$. Then we say that smoking causes cancer if:

$$\pi(b_1 | do a_1) > \pi(b_1 | do a_0)$$

This is clearly an inequality based on probabilities of events rather than variables, but this hasn't hindered the development of BN-based (and hence variable-based) causal modelling: In the above example, one could equally argue that not smoking increases the probability of not developing lung cancer, and hence:

$$\pi(b_0 | do a_0) > \pi(b_0 | do a_1)$$

Putting together a set of such inequalities allows one to express our initial statement in terms of variables rather than events.

The logical development from a BN which describes the relationships between (measurement) variables and whose ordering may be temporal (or follow some other rule), is the Causal Bayesian Network (CBN) [41] whose ordering is causal and whose topology adapts easily to intervention or manipulation. Dawid in [12] extends these ideas to causal modelling in a decision-theoretic framework (replacing the DAG of the BN with the DAG of an Influence Diagram). Lauritzen and Richardson [32] have further extended graphical-model-based causal analysis by applying it to Chain Graphs.

As I have already implied, the idea of a cause and an effect really suggests an event-based approach to causal analysis, and an obvious candidate for this among graphical models is the Decision Tree. It is straightforward to think of a causal manipulation as the making of some decision, and also as such manipulations often induce some asymmetry in a problem, some form of tree would appear to be an ideal representation. Trees have an advantage (one that we can carry forward to CEGs) in that we can choose the level of detail we include in our representation – this will depend on how much we know/understand about the system and also what we intend to *do* to it. Problems arise however in producing a tree if our elicited explanation of the process is not a description of how things happen, or we do not know the order in which things happen. Also, the ordering of our elicited

tree may not be the most appropriate for analysing the effects of the manipulation we are intending to make. If the explanation of the process is not a description of how things happen, this may push the analyst towards using a BN, but analogously with the tree, the topology of the *idle* (or unmanipulated) BN may not lead easily to that of the appropriate CBN.

Shafer [49] has led the way in causal analysis on trees (but see also [46]), but as noted in section 1.2, it is very difficult to extract the conditional independence properties of a system from a tree, and as Pearl [40] has demonstrated, these are very useful for the causal analyst.

In [45], Smith and Riccomagno develop the ideas of causal analysis on CEGs from that of causal analysis on trees, much as the original conception of a CEG was developed through looking at the topology of trees. My approach to the topic is rather different: Anything that we observe about a system or do to a system will change the topology of a graphical representation of that system (see chapter 4), so I come to causal analysis on CEGs through looking at what happens to the topology of a CEG when we observe an event, and considering therefore what might happen to the topology if we manipulate to an event. I believe that conditioning and manipulating are in fact very closely related and that the process of updating our beliefs following a manipulation is necessarily very similar to that which happens following observation of an event. In his creation of a *do calculus* governing the analysis of causal effects, I believe Pearl [41] obscures this affinity. Dawid [12] believes such a *do calculus* is unnecessary, and argues that the tools used in the analysis of probabilistic (as opposed to causal) DAGs are sufficient for causal analysis also. Dawid does choose however to augment his DAGs with additional edges and parentless vertices when undertaking causal analysis, which I believe tends to obscure the closeness of the connection with conditioning on an event. Lauritzen [30] also uses augmented graphs which he calls *intervention graphs*.

As BNs have become the favoured representation for what might be described as *idle* systems, so they have also become (as CBNs) the favoured representation for use with causal analysis. For this reason, where I compare my analysis of CEGs with the analysis of other authors, I focus on analysis on BNs, and in particular the ideas of Pearl [41][42] and how cause relates to control. Note also that this chapter is not based on [60], as the theory herein has developed in tandem with the idea that one can explain all aspects of a model by looking at the $w_0 \rightarrow w_\infty$ paths of the CEG. In this way also, my account of the causal analysis of CEGs diverges considerably from that in [45].

5.2 Conditioning and manipulating

As indicated in section 5.1, I believe that the acts of conditioning on and manipulating to an event are closely related, and I use some of this chapter to provide evidence for this view. I start with a very simple symmetric problem which can be represented both as a BN and as a CEG. So consider the BN in Figure 1, where A, B, C are binary variables.

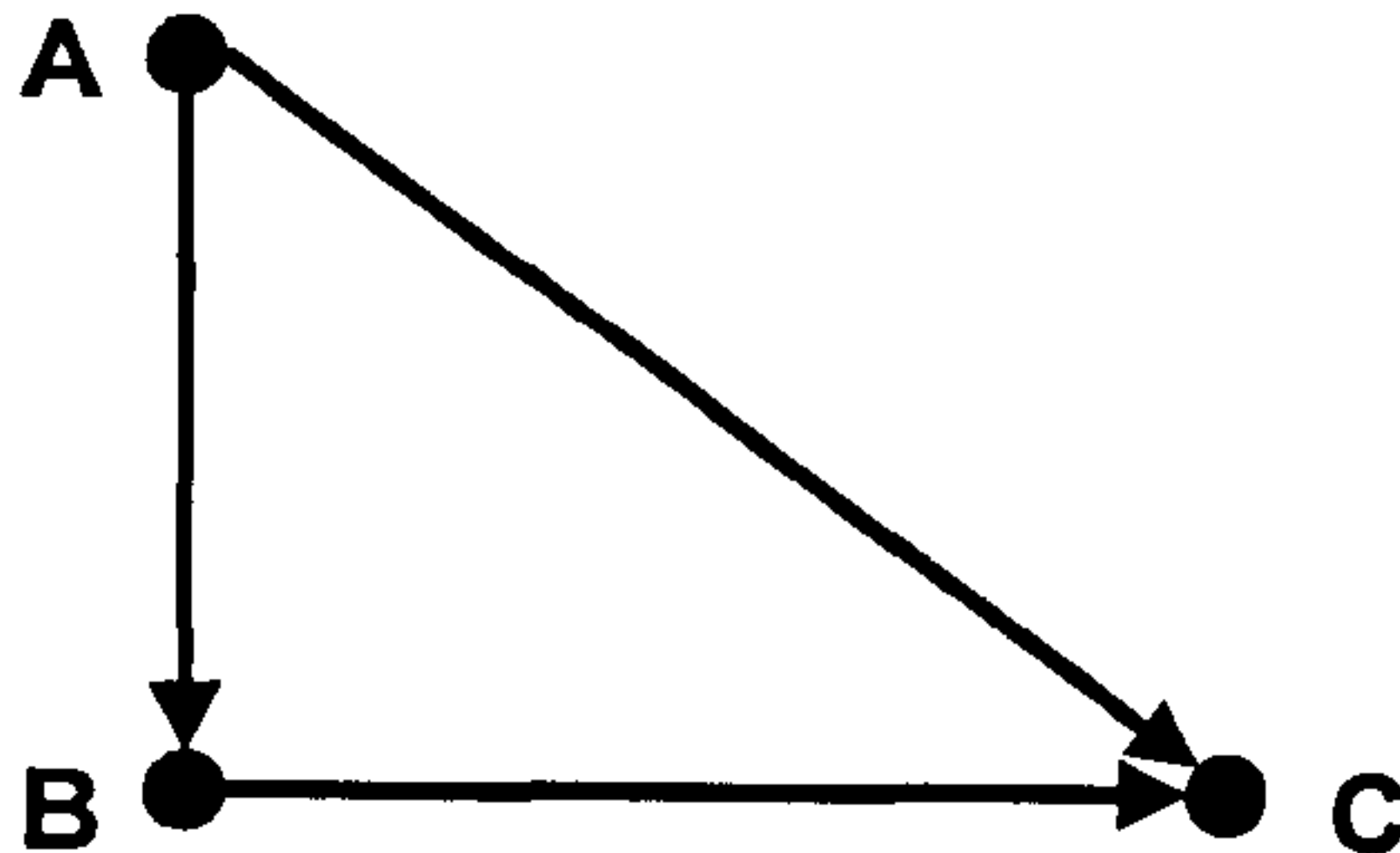


Figure 1: Bayesian Network with three binary variables

I wish to consider the manipulation whereby the variable B is set to a value corresponding to the outcome b_0 , and the effect on the variable C of this manipulation. Pearl would call this intervention $do\ b_0$, and the effects of the intervention are given by expression (3.11) from [41]:

$$P(x_1, \dots, x_n \mid \hat{x}'_i) = \begin{cases} \frac{P(x_1, \dots, x_n)}{P(x'_i \mid pa_i)} & \text{if } x_i = x'_i \\ 0 & \text{if } x_i \neq x'_i \end{cases} \quad (5.2.1)$$

Here X_1, \dots, X_n are a sequence of criterion variables; x_1, \dots, x_n and x'_i are a set of values of these variables; and $P(x_1, \dots, x_n \mid \hat{x}'_i)$ is the probability of achieving this set of values having manipulated X_i to the value x'_i . $P(x'_i \mid pa_i)$ is the probability of X_i taking the value x'_i given that its parents among X_1, \dots, X_n take values in the set x_1, \dots, x_n .

So here $X_1, X_2, X_3 \equiv A, B, C$ and x'_2 corresponds to b_0 . If we consider values x_1, x_2, x_3 corresponding to $a\ b_0\ c$, then $P(x_1, \dots, x_n \mid \hat{x}'_i)$ becomes $\pi(a\ b_0\ c \mid do\ b_0)$. Now if x'_2 corresponds to b_0 then $X_2 \equiv B$ and the only parent of X_2 among X_1, X_2, X_3 is A , so $P(x'_i \mid pa_i)$ becomes $\pi(b_0 \mid a)$, and:

$$\pi(a\ b_0\ c \mid do\ b_0) = \frac{\pi(a\ b_0\ c)}{\pi(b_0 \mid a)}$$

Hence:

$$\begin{aligned} \pi(c_0 \mid do\ b_0) &= \sum_a \left[\frac{\pi(a\ b_0\ c_0)}{\pi(b_0 \mid a)} \right] \\ &= \sum_a \left[\frac{\pi(a) \pi(b_0 \mid a) \pi(c_0 \mid a\ b_0)}{\pi(b_0 \mid a)} \right] \\ &= \sum_a \pi(a) \pi(c_0 \mid a\ b_0) \neq \pi(c_0 \mid b_0) \end{aligned}$$

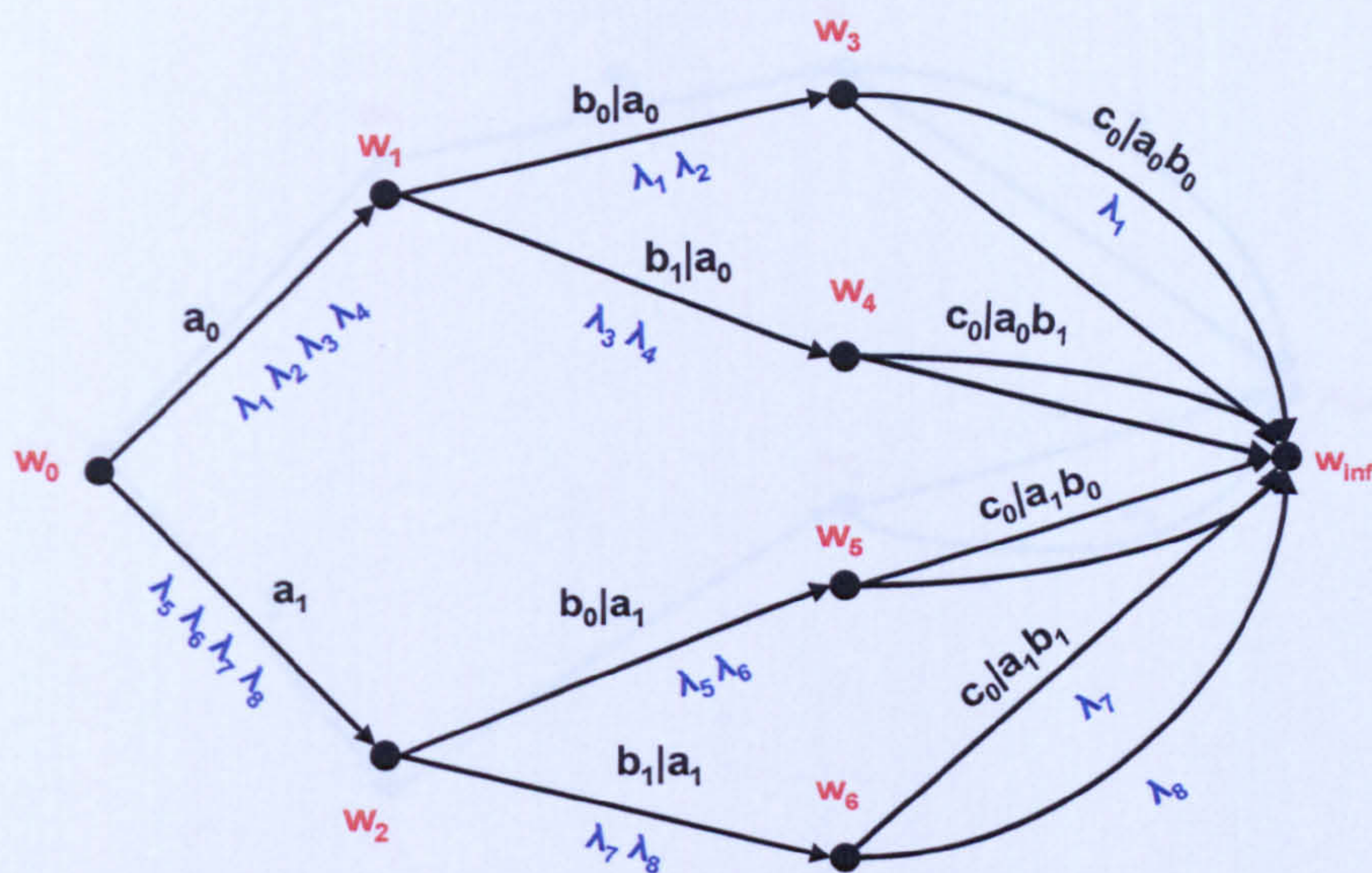


Figure 2: CEG corresponding to BN in Figure 1

Consider also the CEG in Figure 2 which represents the same model, and the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_5 \cup \lambda_6$. Following the argument laid out in chapter 2, we can label this event b_0 , since it is the union of all atomic events that pass through an edge labelled b_0 . Then $G_\Lambda(C)$ (Definition 25 from section 4.6) is as in Figure 3. Suppose we have observed $\Lambda = b_0$; then we can update the CEG edge-probabilities using any of the algorithms from chapter 4. Using that from sections 4.2 and 4.4, we can write $\Lambda = \Lambda(w_3) \cup \Lambda(w_5)$, so steps (1) and (2) give us $G_\Lambda(C)$ (reproduced in Figure 4 with edge-labels for convenience).

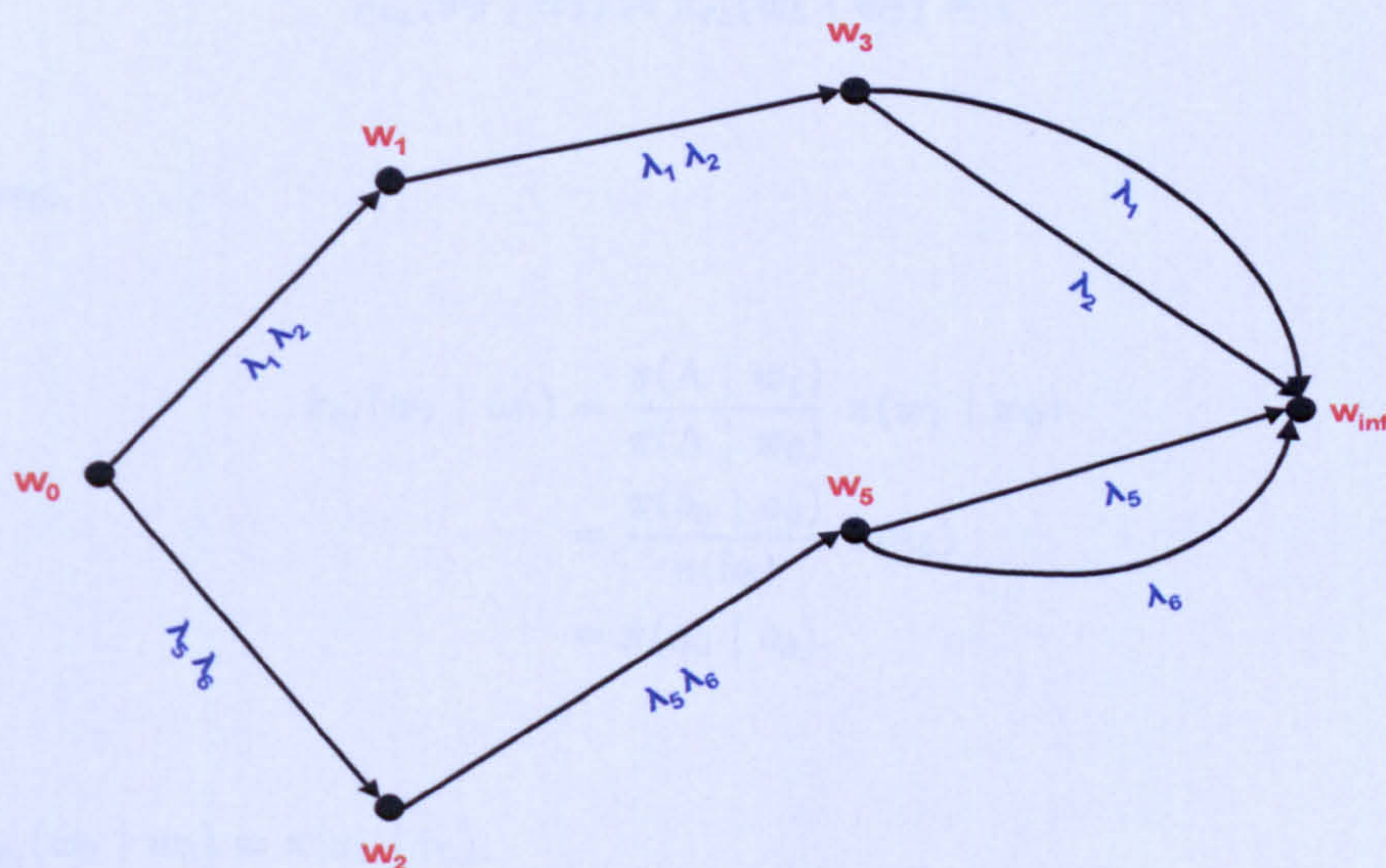


Figure 3: $G_\Lambda(C)$ for $\Lambda = b_0$

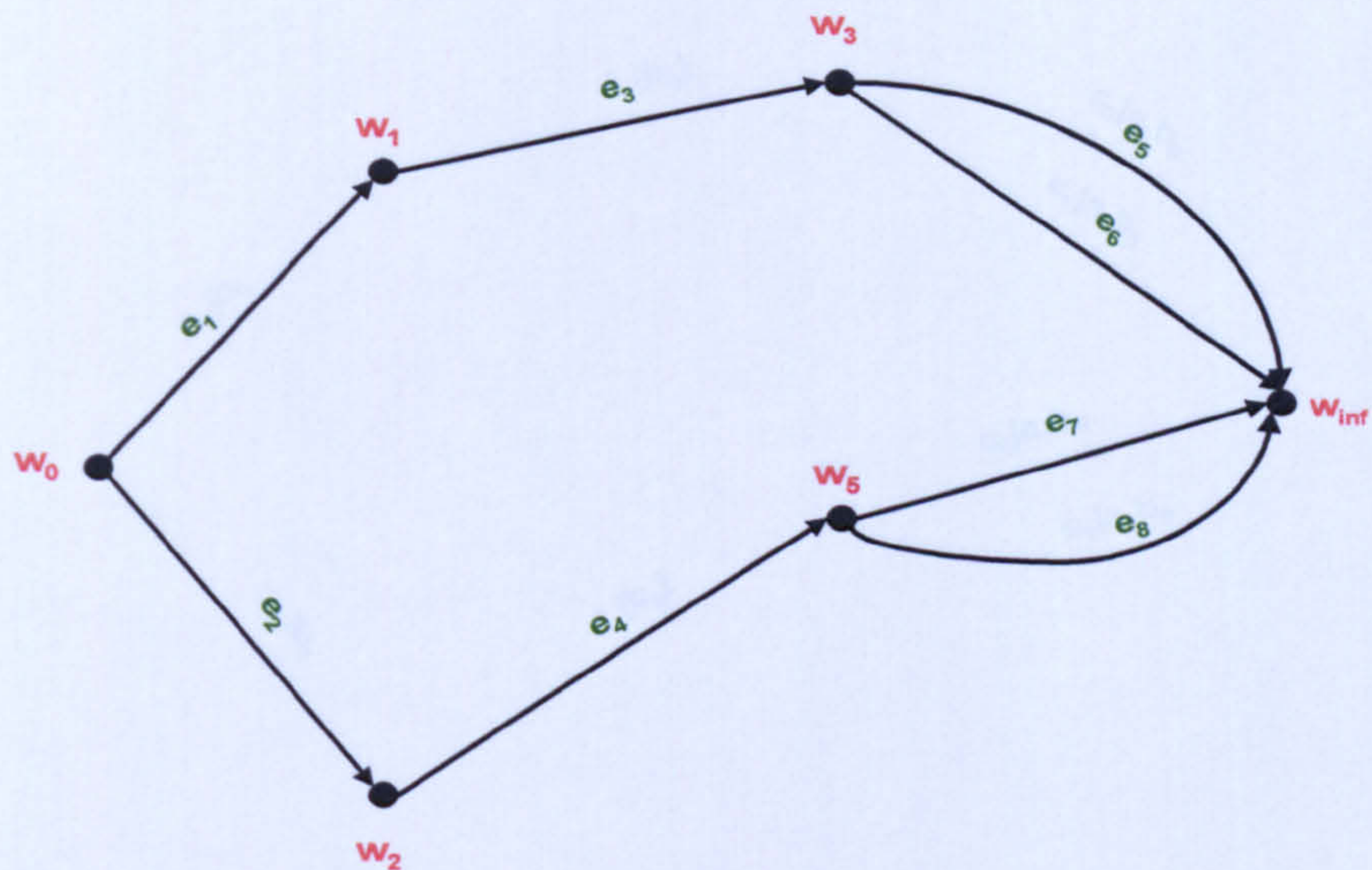


Figure 4: $G_\Lambda(C)$ with edge-labels

Step (3) gives:

$$\hat{\pi}_{e_5}(w_\infty \mid w_3) = \pi_{e_5}(w_\infty \mid w_3) = \pi(c_0 \mid a_0 b_0)$$

Similarly for $\hat{\pi}_{e_6}(w_\infty \mid w_3)$, $\hat{\pi}_{e_7}(w_\infty \mid w_5)$ and $\hat{\pi}_{e_8}(w_\infty \mid w_5)$.

Step (5C) gives:

$$\hat{\pi}_{e_3}(w_3 \mid w_1) = \hat{\pi}_{e_4}(w_5 \mid w_2) = 1$$

Step (5) gives:

$$\begin{aligned} \hat{\pi}_{e_1}(w_1 \mid w_0) &= \frac{\pi(\Lambda \mid w_1)}{\pi(\Lambda \mid w_0)} \pi(w_1 \mid w_0) \\ &= \frac{\pi(b_0 \mid a_0)}{\pi(b_0)} \pi(a_0) \\ &= \pi(a_0 \mid b_0) \end{aligned}$$

Similarly $\hat{\pi}_{e_2}(w_2 \mid w_0) = \pi(a_1 \mid b_0)$.

Step (6) gives us C_Λ as in Figure 5 (note new labelling of positions).

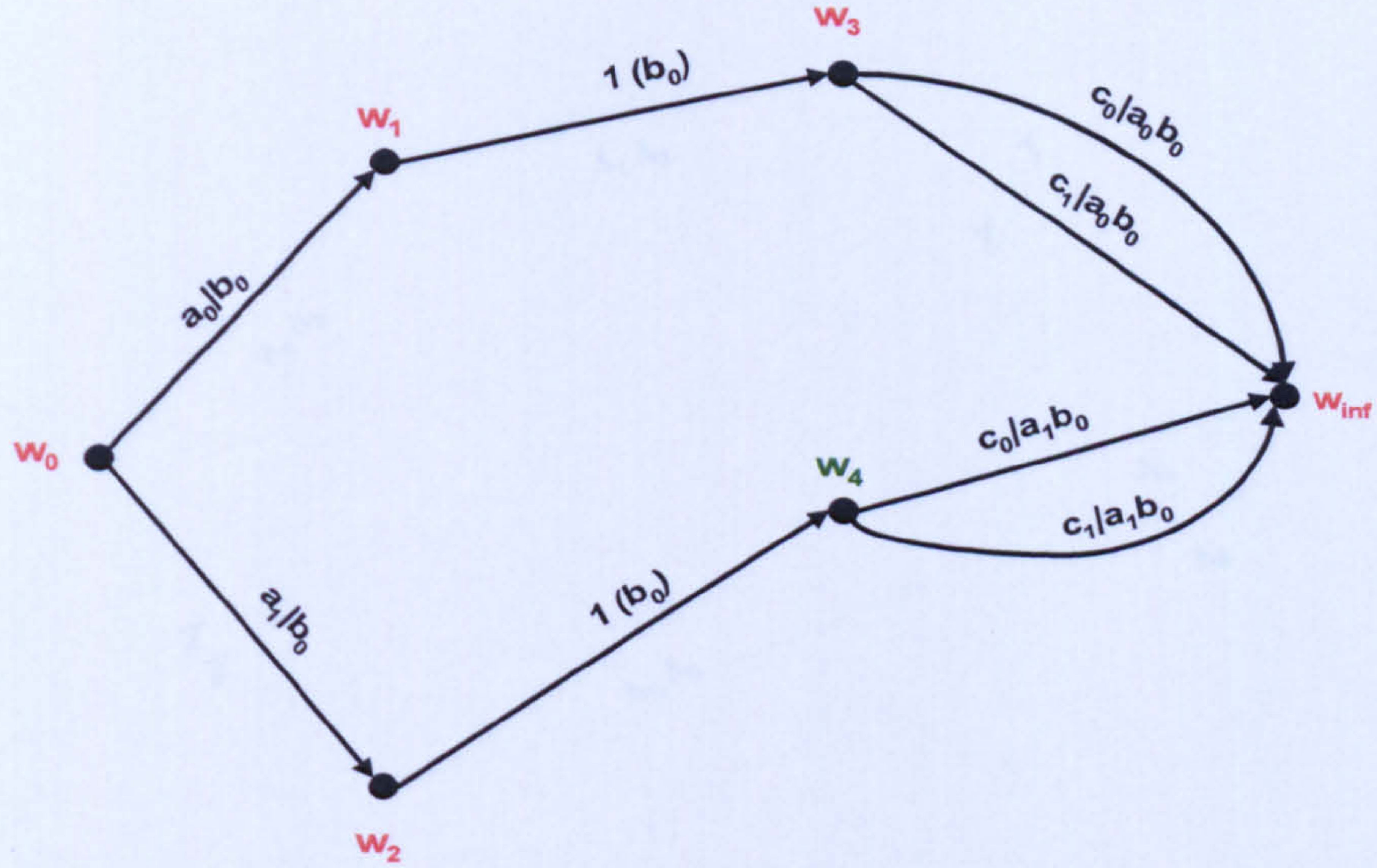


Figure 5: C_Λ for $\Lambda = b_0$

Now suppose instead of conditioning on Λ , we **manipulate** the CEG via the intervention $do \Lambda = do b_0$. Clearly our manipulated CEG $C_{do\Lambda}$ (which will be formally defined in section 5.5.1) must, like C_Λ , be derivable from the graph $G_\Lambda(C)$.

Consider again expression (5.2.1). This gives:

$$\begin{aligned}
 \pi(a_0 b_0 c_0 \mid do b_0) &= \frac{\pi(a_0 b_0 c_0)}{\pi(b_0 \mid a_0)} \\
 &= \frac{\pi(\lambda_1)}{\pi(b_0 \mid a_0)} \\
 &= \frac{\pi(a_0) \pi(b_0 \mid a_0) \pi(c_0 \mid a_0 b_0)}{\pi(b_0 \mid a_0)} \\
 &= \pi(a_0) \pi(c_0 \mid a_0 b_0)
 \end{aligned}$$

But letting probabilities in $C_{do\Lambda}$ be denoted $\hat{\pi}$, we get:

$$\begin{aligned}
 \pi(a_0 b_0 c_0 \mid do b_0) &= \hat{\pi}(a_0 b_0 c_0) \\
 &= \hat{\pi}_{e_1}(w_1 \mid w_0) \hat{\pi}_{e_3}(w_3 \mid w_1) \hat{\pi}_{e_5}(w_\infty \mid w_3)
 \end{aligned}$$

and as the criterion variable A is upstream of the criterion variable B , *doing* b_0 can have no effect on A , so $\hat{\pi}_{e_1}(w_1 \mid w_0) = \pi(a_0)$. Hence:

$$\begin{aligned}
 \pi(a_0 b_0 c_0 \mid do b_0) &= \pi(a_0) \times 1 \times \hat{\pi}_{e_5}(w_\infty \mid w_3) \\
 \Rightarrow \hat{\pi}_{e_5}(w_\infty \mid w_3) &= \pi(c_0 \mid a_0 b_0)
 \end{aligned}$$

Putting these edge-probabilities onto $G_\Lambda(C)$, we get $C_{do\Lambda}$ as in Figure 6:

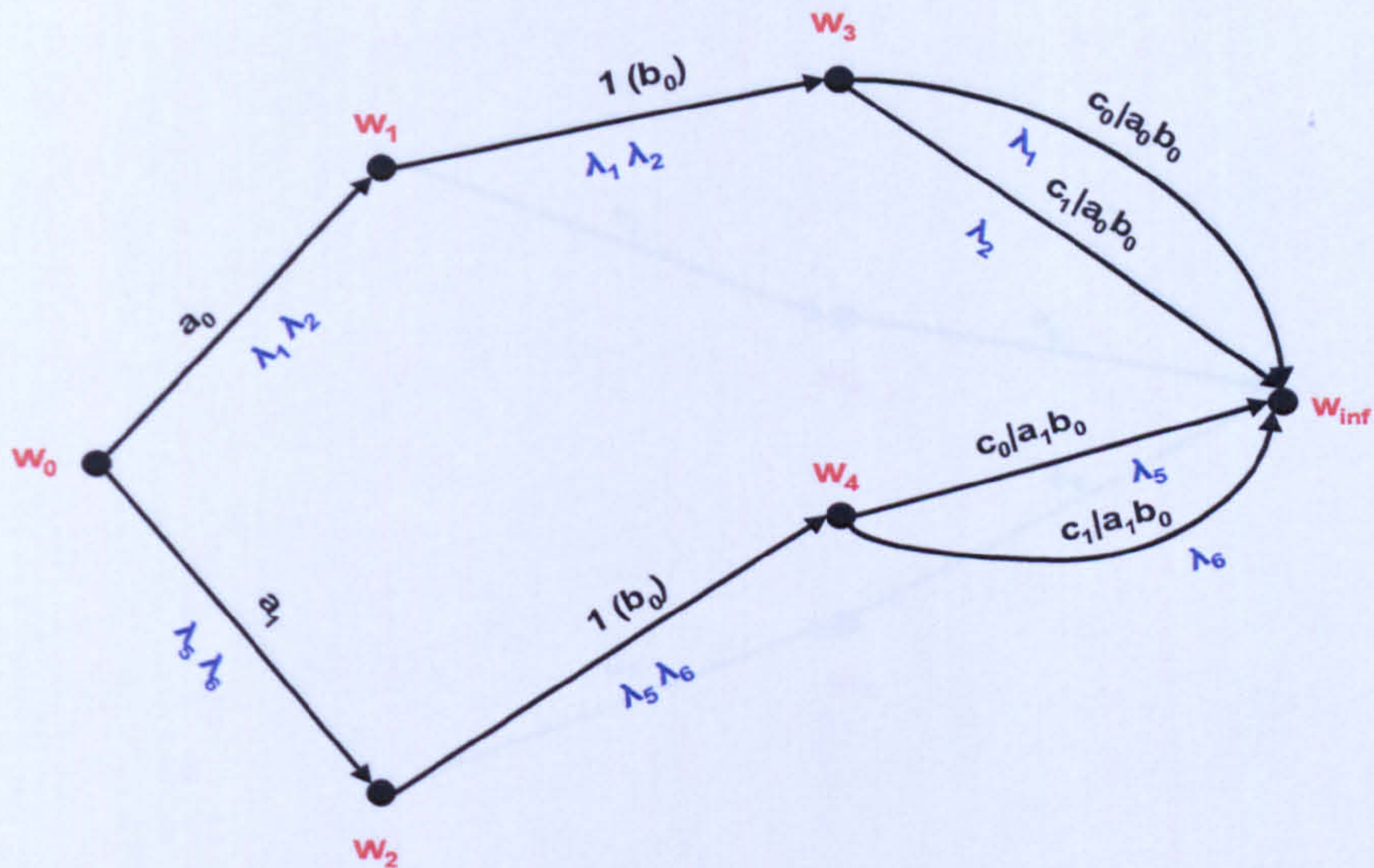


Figure 6: $C_{do\Lambda}$ for $\Lambda = b_0$

Check:

$$\begin{aligned}
 \pi(c_0 \mid do\ b_0) &= \hat{\pi}(c_0) \\
 &= \hat{\pi}(\lambda_1) + \hat{\pi}(\lambda_5) \\
 &= \sum_a \pi(a) \pi(c_0 \mid a\ b_0)
 \end{aligned}$$

which agrees with the expression obtained using the BN in Figure 1.

Note that conditioning on Λ and manipulating to Λ have here given us CEGs C_Λ and $C_{do\Lambda}$ with the same topology, but different edge-probabilities on this structure.

5.3 Defining a manipulation

The example in the previous section poses an interesting question. Here, once we have removed edges which do not lie on a $w_0 \rightarrow w_\infty$ path $\lambda_j \in \Lambda$, all edges except those manipulated (ie. $e(w_1, w_3)$ and $e(w_2, w_5)$ using the position-labelling of Figure 2) retain their original probabilities, unlike in the CEG C_Λ . Is this generally the case?

To be able to answer this question, we need to define what we mean by manipulating to an event Λ when Λ is intrinsic (Definition 26 in section 4.6). The preceding example involves the manipulation of a single criterion variable and it is fairly clear what we mean, especially as it coincides with Pearl's definition of manipulation of a BN. But consider the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_5$. $G_\Lambda(C)$ for this event is given in Figure 7:

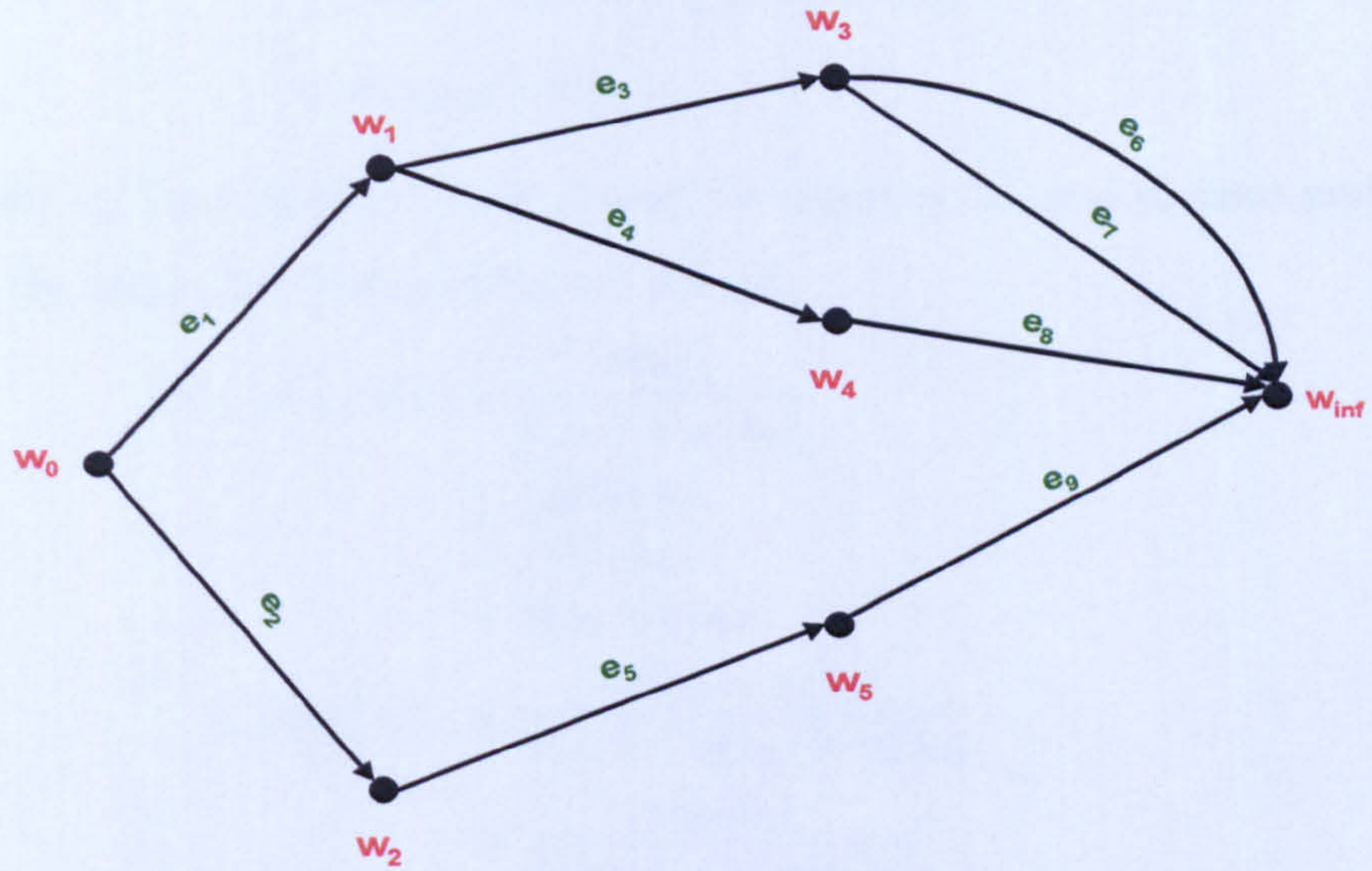


Figure 7: $G_{\Lambda}(C)$ for the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_5$

Now, if we manipulate the CEG by *doing* Λ , it is clear that $\hat{\pi}_{e_8}(w_{\infty} \mid w_4) = \hat{\pi}_{e_5}(w_5 \mid w_2) = \hat{\pi}_{e_9}(w_{\infty} \mid w_5) = 1$, but how do we assign the other probabilities?

Well, we can **define** manipulation to Λ by stating that these retain their original probabilities.

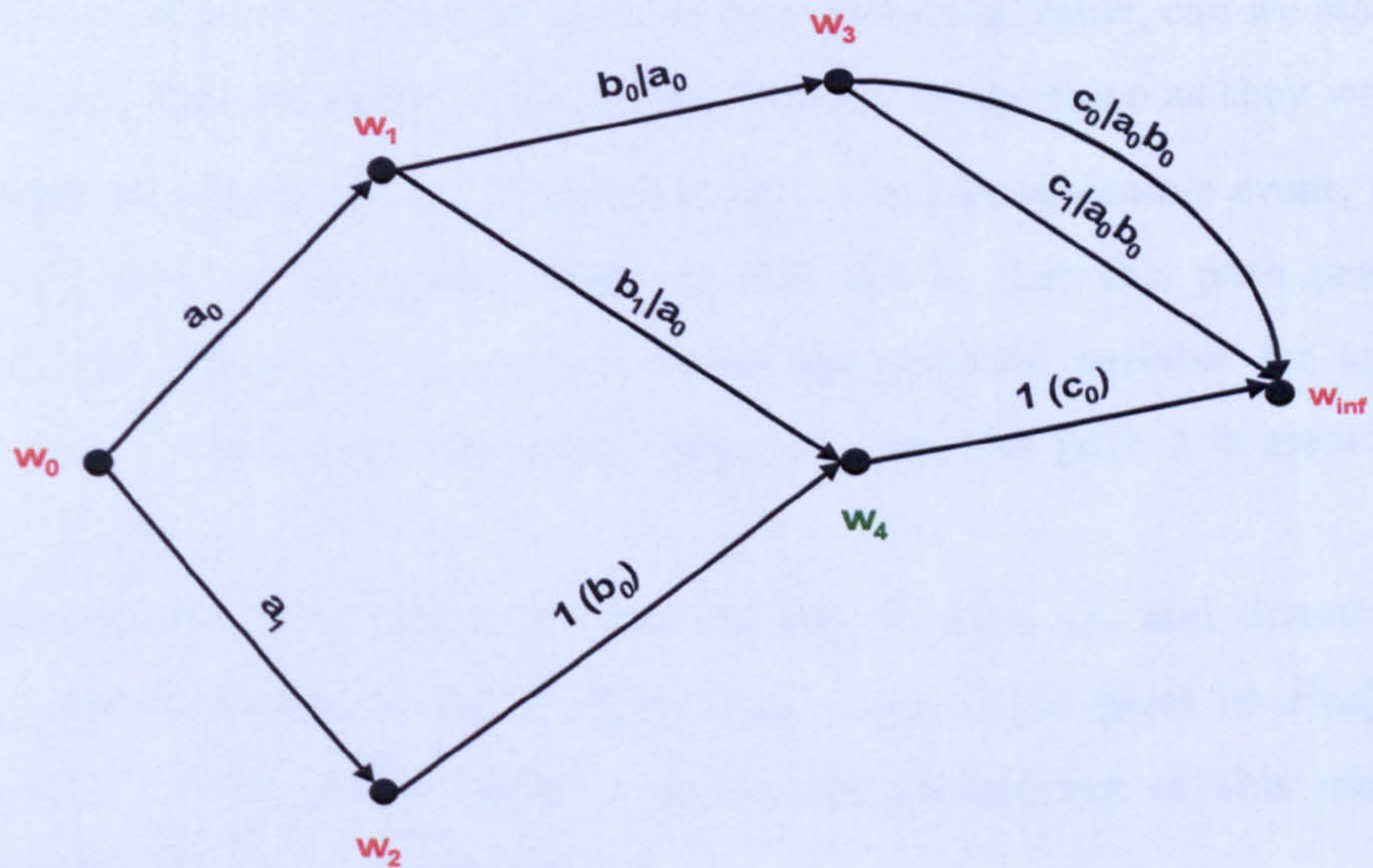


Figure 8: $C_{do\Lambda}$ for the event $\Lambda = \lambda_1 \cup \lambda_2 \cup \lambda_3 \cup \lambda_5$

So we have $C_{do\Lambda}$ as in Figure 8, and for example:

$$\begin{aligned} \pi(b_0 \mid do \Lambda) &= \sum_{a,c} \pi(a b_0 c \mid do \Lambda) \\ &= \sum_{a,c} \hat{\pi}(a b_0 c) \end{aligned}$$

$$\begin{aligned}
&= \pi(a_0) \pi(b_0 \mid a_0) \sum_c \pi(c \mid a_0 b_0) + \pi(a_1) \times 1 \times 1 \\
&= \pi(a_0 b_0) + \pi(a_1)
\end{aligned}$$

If we consider C_Λ for this event, then clearly the edges e_5 , e_8 and e_9 have probability 1.

Also, using the algorithm from section 4.7, we get:

$$\begin{aligned}
\hat{\pi}_{e_6}(w_\infty \mid w_3) &= \frac{\pi(\lambda_1)}{\pi(\lambda_1) + \pi(\lambda_2)} \\
&= \frac{\pi(a_0 b_0 c_0)}{\pi(a_0 b_0)} \\
&= \pi(c_0 \mid a_0 b_0) \\
\hat{\pi}_{e_3}(w_3 \mid w_1) &= \frac{\pi(\lambda_1) + \pi(\lambda_2)}{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3)} \\
&= \frac{\pi(a_0 b_0)}{\pi(a_0 b_0) + \pi(a_0 b_1 c_0)} \\
&\neq \pi(b_0 \mid a_0) \\
\hat{\pi}_{e_1}(w_1 \mid w_0) &= \frac{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3)}{\pi(\lambda_1) + \pi(\lambda_2) + \pi(\lambda_3) + \pi(\lambda_5)} \\
&= \frac{\pi(a_0 b_0) + \pi(a_0 b_1 c_0)}{\pi(a_0 b_0) + \pi(a_0 b_1 c_0) + \pi(a_1 b_0 c_0)} \\
&\neq \pi(a_0)
\end{aligned}$$

But is our new definition consistent with Pearl's definitions? That is, if *doing* Λ can be expressed as the setting of a criterion variable to a particular value, can we state that all edge-probabilities that we assign to $G_\Lambda(C)$ are either 1 or the same as they were in C ?

Let us consider expression (5.2.1). Note that (x_1, \dots, x_n) is an atomic event, and hence in our CEG C a $w_0 \rightarrow w_\infty$ path, which we will call λ . Let this path pass through a set of positions $\{w_0, w_1, \dots, w_{n-1}, w_\infty\}$ where the criterion variable for w_k is X_{k+1} ($k = 0, 1, \dots, n-1$), and where the edge $e(w_k, w_{k+1})$ on this path λ is associated with the value $X_{k+1} = x_{k+1}$.

In the expression for $P(x'_i \mid pa_i)$, we can let $pa_i = \Lambda(w_{i-1})$, and denote the edge leaving w_{i-1} corresponding to $X_i = x'_i$ by $e(w_{i-1}, w_i)$. This gives us $P(x'_i \mid pa_i) = \pi(\Lambda(e(w_{i-1}, w_i)) \mid \Lambda(w_{i-1})) = \pi_e(w_i \mid w_{i-1})$, the probability of this edge on the path λ (using Result 3 from section 3.6).

So expression (5.2.1) gives us that:

$$\pi(\lambda \mid do X_i = x'_i) = \hat{\pi}(\lambda) = \frac{\pi(\lambda)}{\pi_e(w_i \mid w_{i-1})}$$

Now this is certainly consistent with our new definition, but we will also look at Pearl's alternative expression (3.10) (from [41]):

$$P(x_1, \dots, x_n \mid \hat{x}'_i) = \begin{cases} \prod_{j \neq i} P(x_j \mid pa_j) & \text{if } x_i = x'_i \\ 0 & \text{if } x_i \neq x'_i \end{cases} \quad (5.2.2)$$

Using the reasoning employed above, we can write each $P(x_j \mid pa_j)$ as $\pi_e(w_j \mid w_{j-1})$ for some edge $e(w_{j-1}, w_j)$ on path λ , giving:

$$\pi(\lambda \mid do X_i = x'_i) = \hat{\pi}(\lambda) = \prod_{j \neq i} \pi_e(w_j \mid w_{j-1})$$

where the edges $e(w_{j-1}, w_j)$ are the component edges of λ . But clearly:

$$\hat{\pi}(\lambda) = \prod_j \hat{\pi}_e(w_j \mid w_{j-1})$$

and since $\hat{\pi}_e(w_i \mid w_{i-1}) = 1$, we can write:

$$\hat{\pi}(\lambda) = \prod_{j \neq i} \hat{\pi}_e(w_j \mid w_{j-1})$$

So:

$$\prod_{j \neq i} \hat{\pi}_e(w_j \mid w_{j-1}) = \prod_{j \neq i} \pi_e(w_j \mid w_{j-1})$$

We can make this identity hold, simply by letting $\hat{\pi}_e(w_j \mid w_{j-1}) = \pi_e(w_j \mid w_{j-1})$ for all $j \neq i$.

So, if *doing* Λ can be expressed as the the setting of a criterion variable to a particular value, then all edge-probabilities can be made to be either 1 or the same as they were in C , and our definition is consistent with Pearl's.

But events that can be expressed as the the setting of a criterion variable to a particular value constitute the major part of all events used in Causal analysis on BNs, and as we have just seen, producing a manipulated-CEG for this sort of event is almost trivial.

It is worth stressing again here that, at least for intrinsic events, $C_{do \Lambda}$, like C_Λ , can be derived quickly from $G_\Lambda(C)$, but it is a very much simpler task to update the probabilities on $C_{do \Lambda}$ than it is on C_Λ .

Having introduced the idea of manipulated CEGs, we now look at an example where the importance of a precise definition for our manipulation is made abundantly clear.

5.4 Another look at the Machine example

5.4.1 Our initial CEG

We return here to the example first encountered in section 2.6 which is concerned with a fault-monitoring process in a production line (Example 4). For convenience we repeat the description of the process here:

A machine in a production line utilises two replaceable components M and N. Faults in these components do not automatically cause the machine to fail, but do affect the quality of the product, so the machine incorporates an automated monitoring system, which is completely reliable for finding faults in M, but which can detect a fault in N when it is functioning correctly.

In any monitoring cycle, component M is checked first, and there are three initial possibilities:

M, N checked and no faults found (a_0 on the CEG in Figure 9); M checked, fault found, machine switched off (a_1); M checked, no fault found, N checked, fault found, machine switched off (a_2).

If M is found faulty it is replaced and the machine switched back on (position w_1), and N is then checked. N is then either found not faulty (b_0), or faulty and the machine switched off (b_1).

If N is found faulty by the monitoring system, then it is removed and checked (positions w_2 and w_3). There are then three possibilities, **whose probabilities are independent of whether or not component M has been replaced**: N is not in fact faulty, the machine is reset and restarted (c_0); N is faulty, is successfully replaced and the machine restarted (c_1); N is faulty, is not replaced successfully and the machine is left off until the engineer can see it (c_2).

At the time of any monitoring cycle, the quality of the product produced (represented here by a binary criterion variable with d_0 indicating a *bad product* and d_1 indicating a *good product*) is unaffected by the replacement of M unless N is also replaced. It is however dependent on the *effectiveness* of N which depends on its *age*, but also, if it is a **new component**, on the *age* of M; so: $\pi(d_1 \mid M \text{ and } N \text{ replaced}) > \pi(d_1 \mid \text{only } N \text{ replaced}) > \pi(d_1 \mid N \text{ not replaced})$.

This scenario can be represented by the CEG in Figure 9. Figure 10 shows the same CEG with edge-probabilities added.

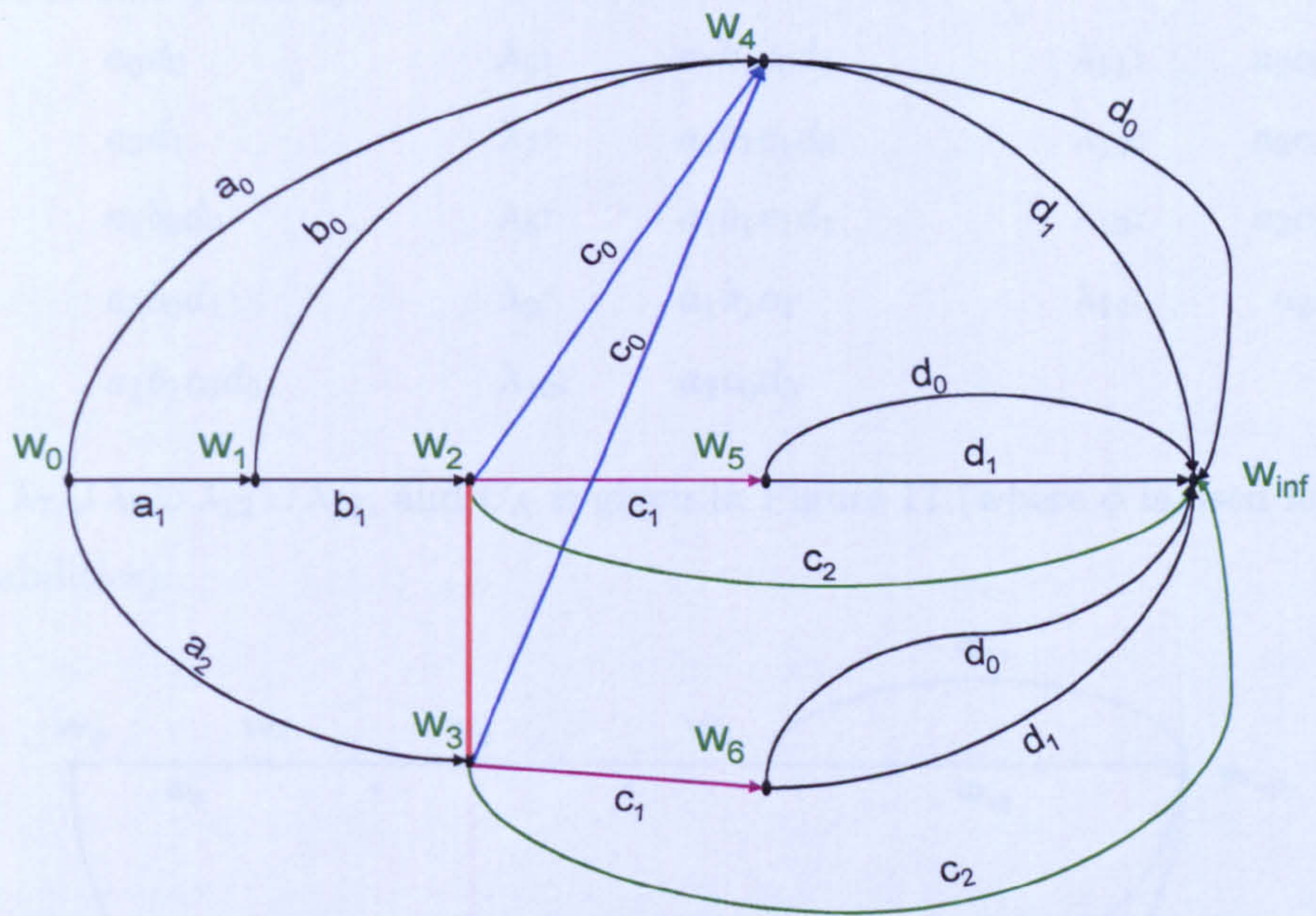


Figure 9: CEG for Machine Example

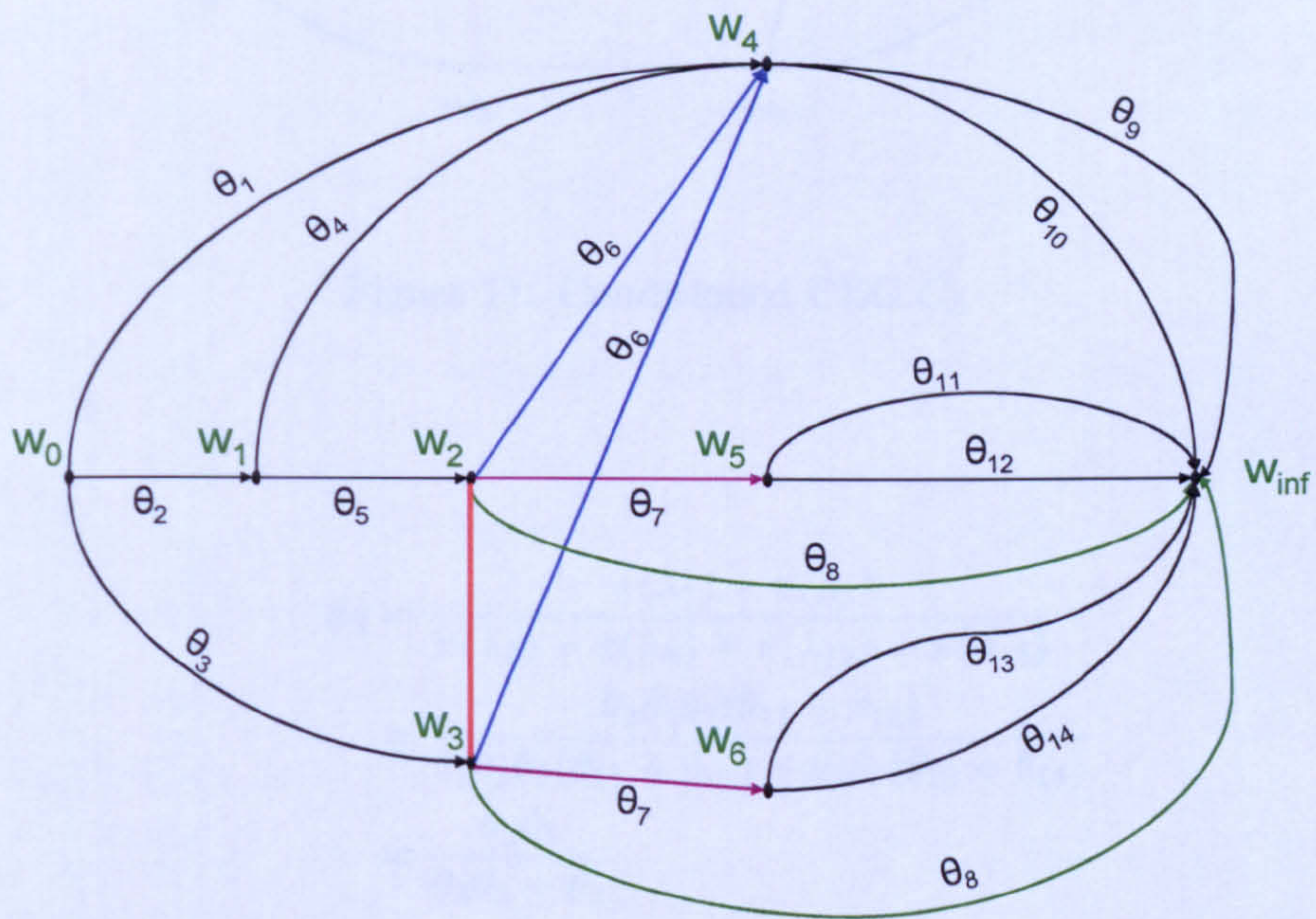


Figure 10: CEG showing probabilities

5.4.2 Conditioning on an event Λ

We look first at simply conditioning on an event $\Lambda = \bigcup_{i \in I} \lambda_i$ (where $\{\lambda_i\}$ is the set of root-to-sink paths in our CEG, and $\{\lambda_i\}_{i \in I}$ is some subset of this set), and use the algorithm from section 4.7 to construct C_Λ , the CEG conditioned on the event Λ . We consider the event $\Lambda = N \text{ is replaced}$, which is intrinsic to our CEG C . We can label our

(where $\hat{\pi}$ denotes a probability in the CEG C_Λ)

$$\begin{aligned}
&= \phi_2 \times 1 \times 1 \times \phi_{12} + \phi_3 \times 1 \times \phi_{14} \\
&= \frac{\theta_2 \theta_5 \theta_{12} + \theta_3 \theta_{14}}{\theta_2 \theta_5 + \theta_3}
\end{aligned}$$

5.4.3 Manipulating to an event Λ

If we now consider the manipulation $do \Lambda = do (N \text{ replaced})$, we immediately discover that, unlike conditioning on Λ , manipulating to Λ is not always a clearly defined process. Heckerman and Shachter in [19] point out that Pearl assumes that in systems which can be represented by BNs, each variable in a model can be manipulated to a set of possible values or left idle. This is clearly a restricted set of the possible interventions – as Dawid so aptly puts it in [11], one can set a patient’s treatment to none by withholding it, wiring his jaw shut or killing him, and these are all different interventions. Dawid makes a similar point about the multiplicity of possible interventions in comments on Lauritzen and Richardsons’ [32]. Interestingly, Heckerman and Shachter, who also follow a decision-theoretic approach, call on Savage [48] in their argument that consequences of an act (effects of an intervention) are **deterministic** functions of the act and the *state of the world*. The uncertainty arrives because we are uncertain about the state of the world.

In the present case, do we mean:

- (a) Replace N whatever (and have an engineer present to ensure that the replacement is successful), or
- (b) Replace N if the monitoring system tells us that N is faulty (and have an engineer present to ensure that the replacement is successful), or
- (c) Replace N only if it is actually faulty (and have an engineer present to ensure that the replacement is successful)?

Now, for symmetric models (those which can be completely specified by a BN), and for manipulations of the form $do C = c_0$ (where C is a *criterion variable* equivalent to one of the vertex-variables in the appropriate BN), we would expect from the previous section the manipulated CEG $C_{do\Lambda}$ to have essentially the same topology as the conditioned CEG C_Λ , but to have *simpler* edge-probabilities in that all edges retain their initial probabilities except the manipulated edges which are given a probability of 1.

Our model is not symmetric, and neither are we considering an atomic manipulation of the form discussed. Hence our $C_{do\Lambda}$ is not going to be as simple as that just described.

Our initial response to the question *What do we mean by do (N replaced)?* is likely to

be (c), but in fact, as we will shortly see, this produces a $C_{do\Lambda}$ further removed in topology from C_Λ than that produced by either (a) or (b). We start instead by looking at (a), which at least superficially gives a $C_{do\Lambda}$ with similar topology to that of C_Λ .

If we replace N whatever (and have an engineer present to ensure that the replacement is successful), then there should essentially be four root-to-sink paths in our manipulated CEG, representing:

- M faulty and replaced, N checked and diagnosis ignored, N replaced successfully, bad product
- M faulty and replaced, N checked and diagnosis ignored, N replaced successfully, good product
- M not faulty, N checked and diagnosis ignored, N replaced successfully, bad product
- M not faulty, N checked and diagnosis ignored, N replaced successfully, good product

Our manipulated CEG $C_{do\Lambda}$ is then given in Figure 12.

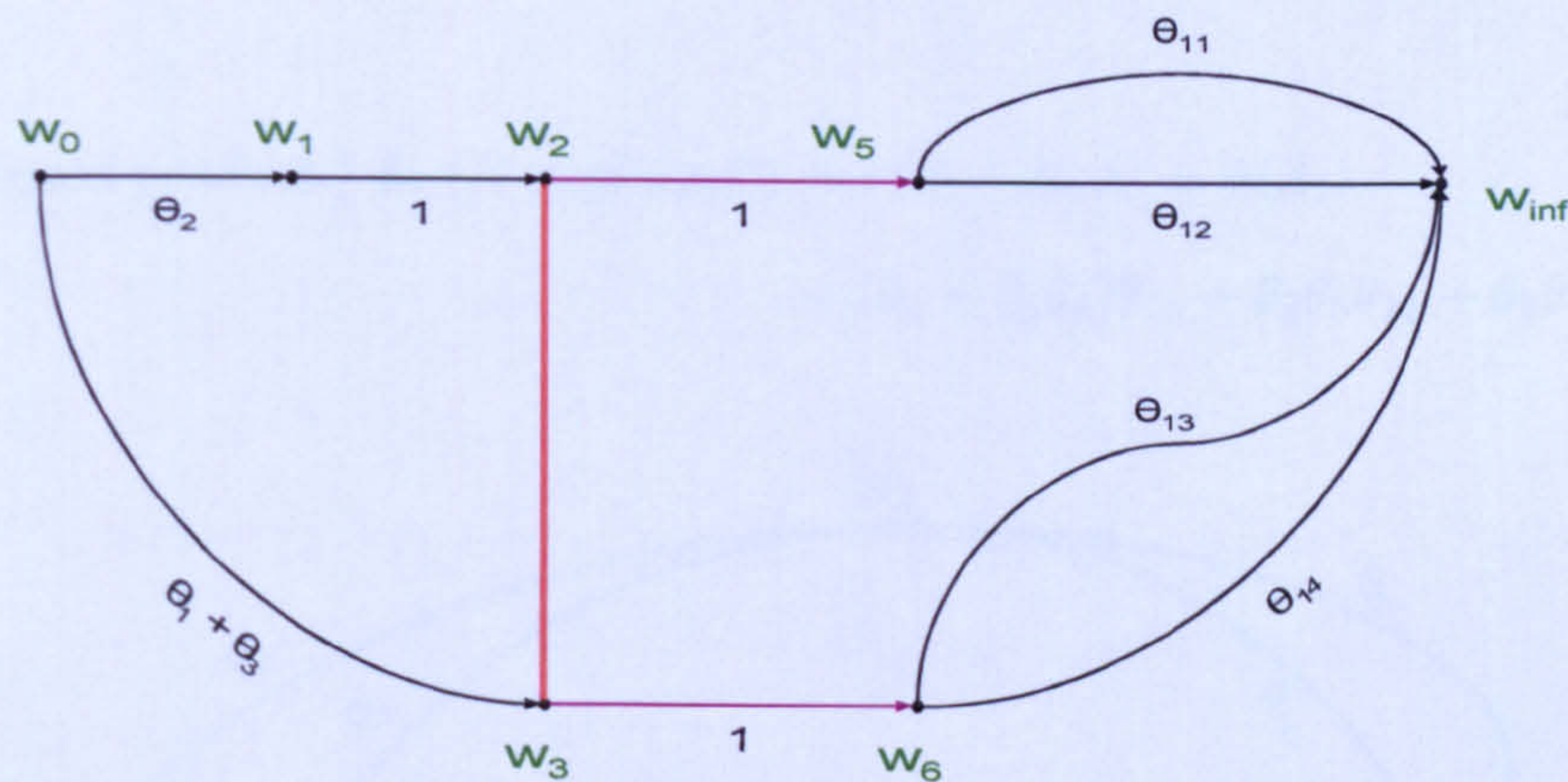


Figure 12: Manipulated CEG $C_{do\Lambda}$ for option (a)

Here, most edges retain their original probabilities, and the edges originally labelled c_1 are given probabilities of 1.

The edge $e(w_1, w_2)$ of course could be thought of as two edges, those originally labelled b_0 (now redirected to w_2) and b_1 , which could retain their original probabilities θ_4 and θ_5 (with $\theta_4 + \theta_5 = 1$). Similarly the edge $e(w_0, w_3)$ could be thought of as two edges, those originally labelled a_0 (now redirected to w_3) and a_2 , which could retain their original probabilities θ_1 and θ_3 . This would make $C_{do\Lambda}$ more *typical* in that all edges retained their original probabilities except manipulated edges, but also less typical in that $C_{do\Lambda}$ would have a very different topology from C_Λ (and also from C). It would also be unnecessarily complex.

Note that:

$$\pi(\text{good product} \mid \text{do}(N \text{ replaced})) = \pi(d_1 \mid \text{do } \Lambda) = \hat{\pi}(d_1)$$

(where $\hat{\pi}$ here denotes a probability in the CEG $C_{\text{do}\Lambda}$)

$$= \theta_2\theta_{12} + (\theta_1 + \theta_3)\theta_{14}$$

a much simpler expression than that for $\pi(\text{good product} \mid N \text{ replaced})$.

Let us now consider option (b). If we replace N when the monitoring system tells us that N is faulty (and have an engineer present to ensure that the replacement is successful), we get the manipulated CEG $C_{\text{do}\Lambda}$ in Figure 13. Here the topology is somewhere between that of C and C_Λ , but it does have all edges retaining their probabilities except manipulated edges. Note however that not all root-to-sink paths in $C_{\text{do}\Lambda}$ utilise a manipulated edge.

Here

$$\pi(\text{good product} \mid \text{do}(N \text{ replaced})) = \pi(d_1 \mid \text{do } \Lambda) = \hat{\pi}(d_1)$$

$$= (\theta_1 + \theta_2\theta_4)\theta_{10} + \theta_2\theta_5\theta_{12} + \theta_3\theta_{14}$$

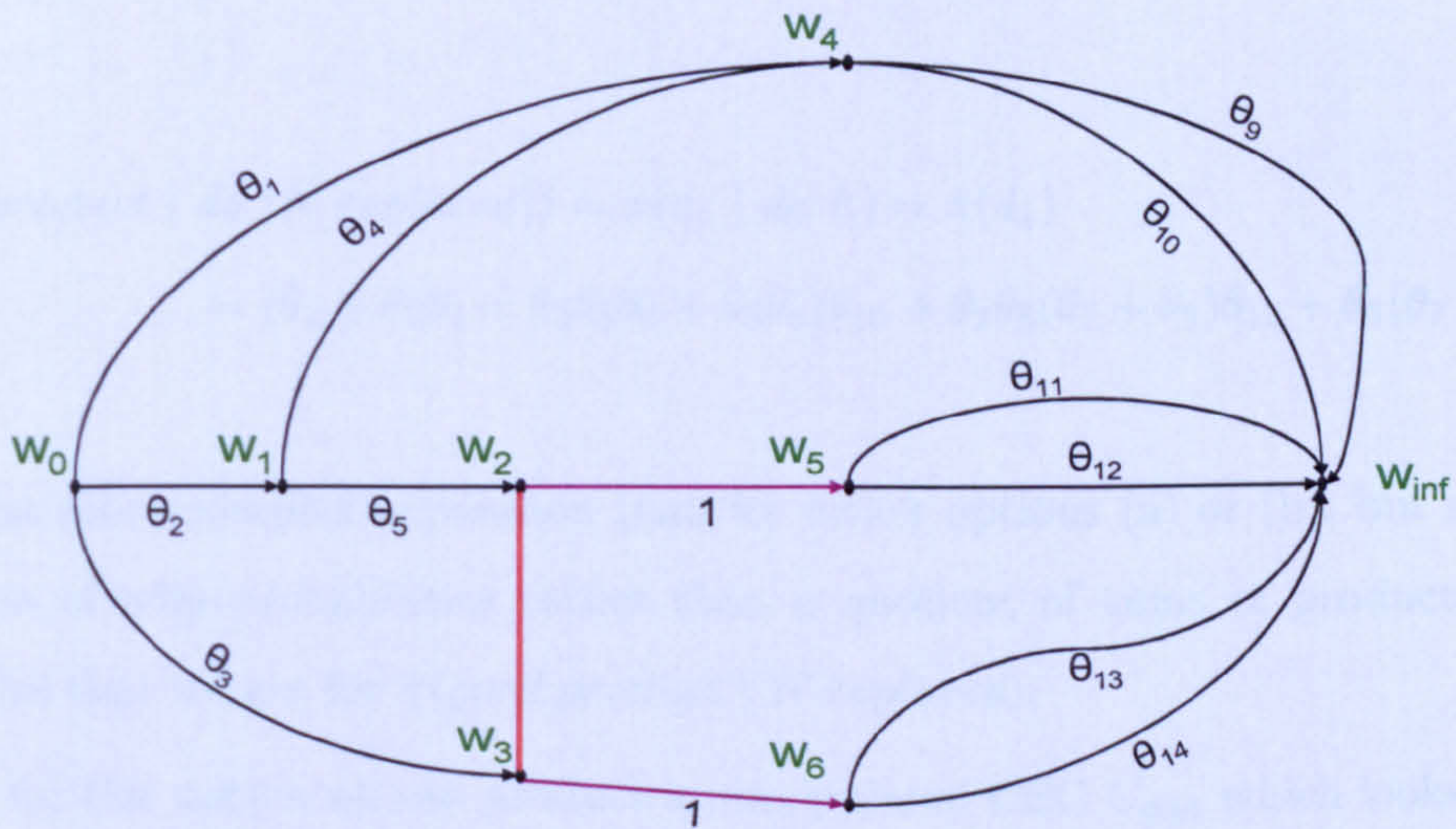


Figure 13: Manipulated CEG $C_{\text{do}\Lambda}$ for option (b)

Lastly we consider option (c). If we replace N only if it is actually faulty (and have an engineer present to ensure that the replacement is successful), then our $C_{\text{do}\Lambda}$ is as in Figure 14.

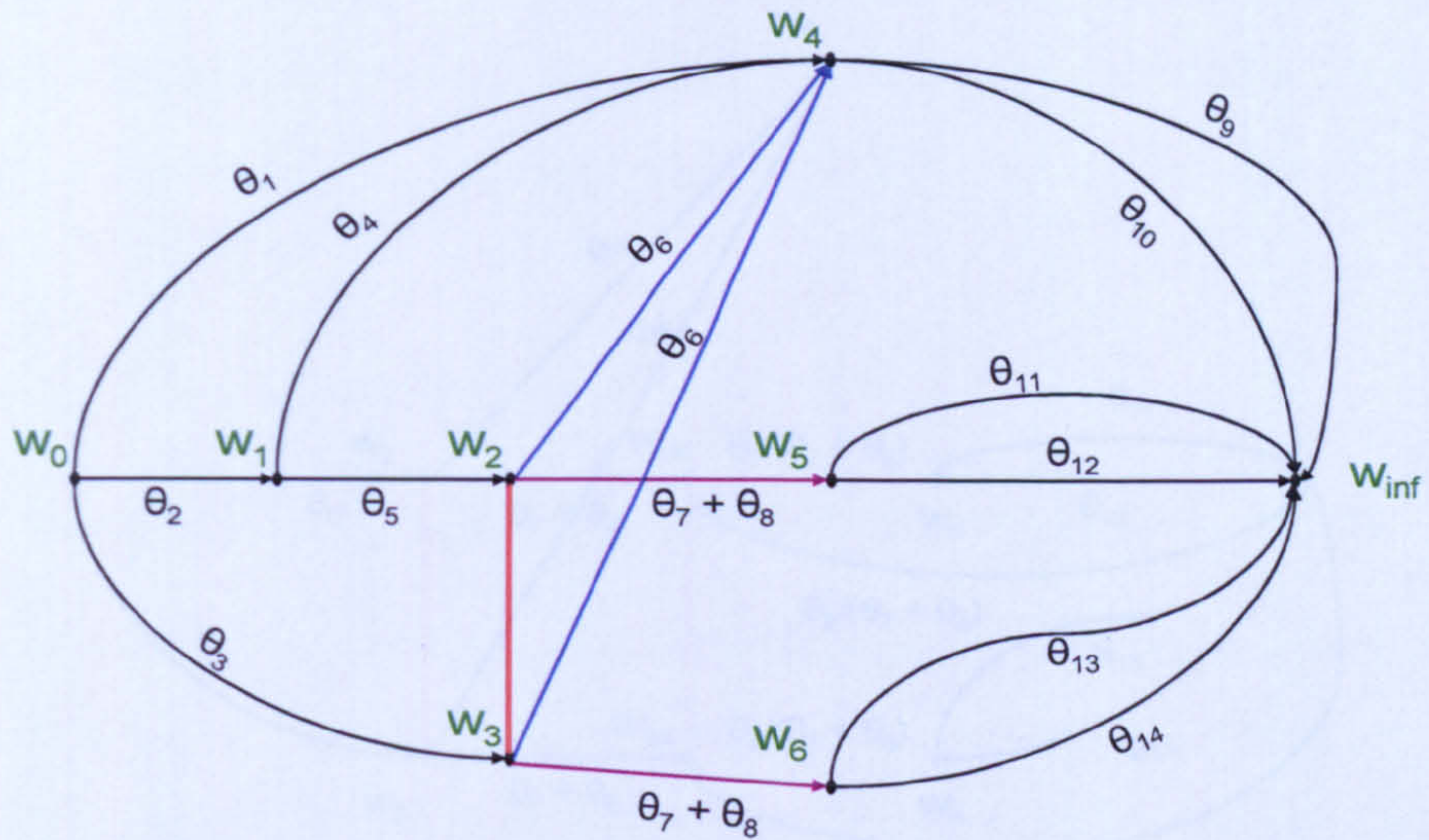


Figure 14: Manipulated CEG $C_{do\Lambda}$ for option (c)

Here the topology is again somewhere between that of C and C_Λ , but closer to that of C . Note that there are no edges with probabilities of 1!

Now

$$\begin{aligned} \pi(\text{good product} \mid do(N \text{ replaced})) &= \pi(d_1 \mid do \Lambda) = \hat{\pi}(d_1) \\ &= (\theta_1 + \theta_2\theta_4 + \theta_2\theta_5\theta_6 + \theta_3\theta_6)\theta_{10} + \theta_2\theta_5(\theta_7 + \theta_8)\theta_{12} + \theta_3(\theta_7 + \theta_8)\theta_{14} \end{aligned}$$

a somewhat more complex expression than for either options (a) or (b), but still a sum of products of edge-probabilities rather than a quotient of sums of products of edge-probabilities that we get for $\pi(\text{good product} \mid N \text{ replaced})$.

Note that for this option we can produce a manipulated CEG $C_{do\Lambda}$ which looks more like what we might typically expect by the simple expedient of changing the structure of the original CEG C ! If we modify C so that at each of w_2, w_3 we have a binary outcome space $\{N \text{ actually faulty}, N \text{ not actually faulty}\}$, and add in extra positions w_{2a}, w_{3a} with outcomes $\{N \text{ replaced successfully}, N \text{ not replaced successfully}\}$, then we get the CEG in Figure 15, and our manipulated CEG $C_{do\Lambda}$ is as in Figure 16. We now have all edges other than manipulated edges retaining their original probabilities, and manipulated edges having probabilities of 1.

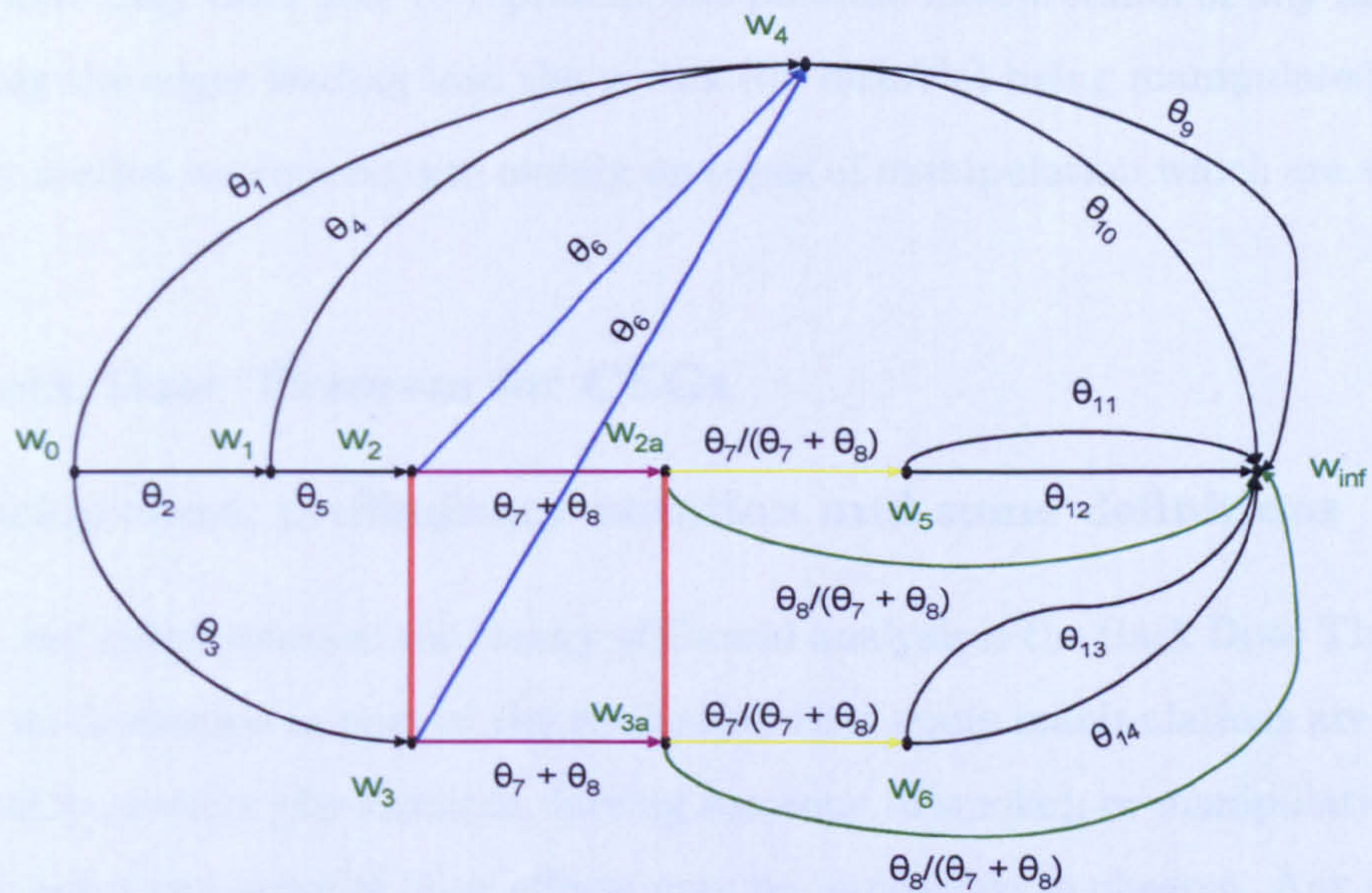


Figure 15: Alternative CEG C

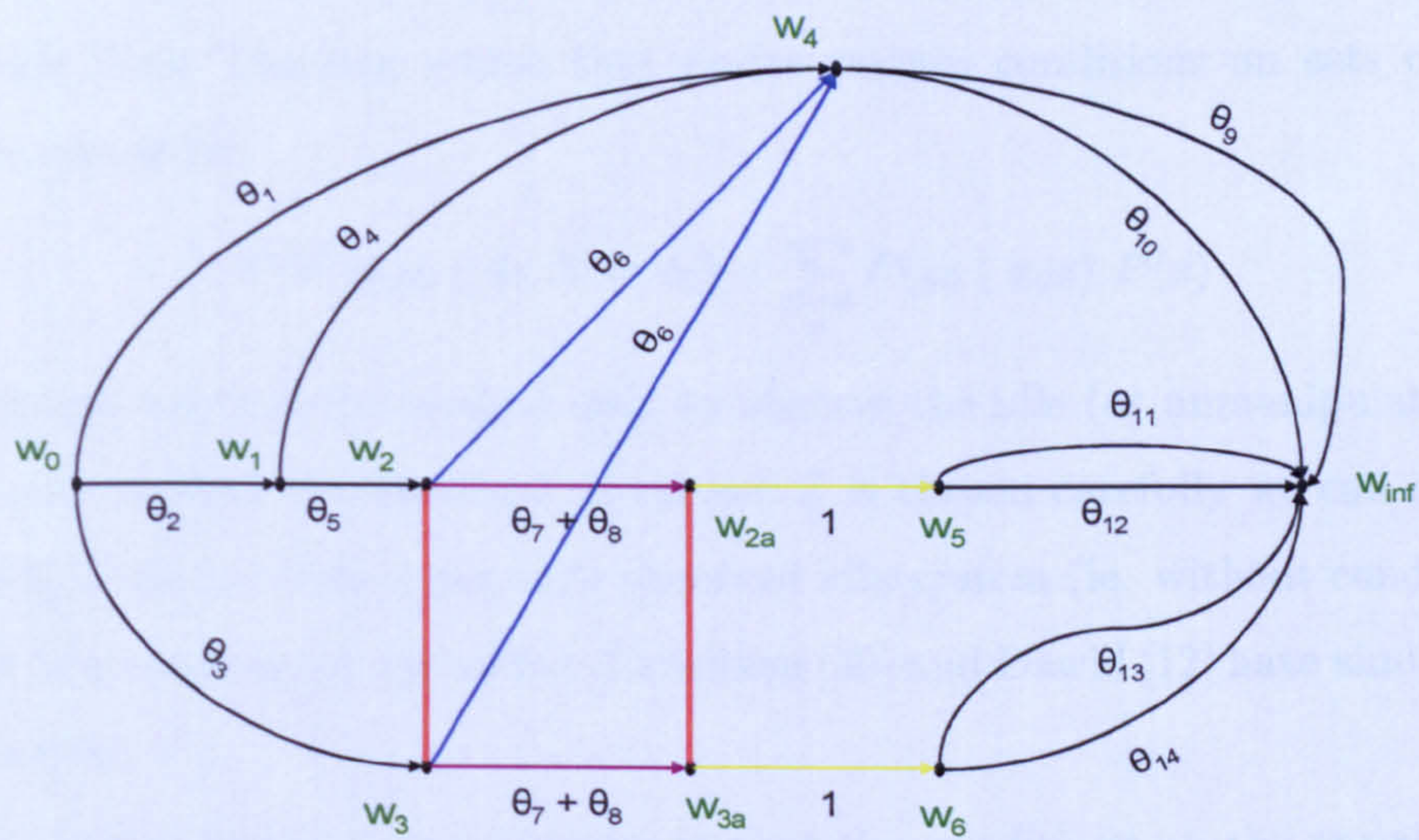


Figure 16: Alternative CEG $C_{do\Lambda}$

We have seen in this section that for asymmetric processes we need to be very careful how we define our manipulations. We have also seen that the topology of the manipulated CEG $C_{do\Lambda}$ depends very much on how we define $do \Lambda$.

Notice also how much more expressive our CEG and our manipulated CEGs are in comparison with BNs and CBNs. We have been able very easily to produce manipulated CEGs for each of our interpretations of the intervention do ($N replaced$), whereas with a BN, not only would we have been unable to represent the initial problem adequately, but

we would have only been able to represent one possible interpretation of any intervention, by removing the edges leading into the vertex (or variable) being manipulated.

In the next section we concentrate mainly on types of manipulation which are very clearly defined.

5.5 A Back Door Theorem for CEGs

5.5.1 Background, preliminary notation and some definitions

One of the key components of the theory of Causal analysis is the Back Door Theorem [41]. This owes its derivation in part to the realisation that many manipulations are impossible or unethical in practice (for example, forcing someone to smoke); or manipulations may be possible to enact but some of their effects may be impossible to observe. Any expressions whereby the effects of a causal manipulation can be expressed using only marginal, joint and standard conditional probabilities will therefore be very useful.

Pearl's Back Door Theorem states that under certain conditions on sets of variables X, Y, Z , we can write:

$$P(Y = y_0 \mid do X = x_0) = \sum_z P(y_0 \mid x_0 z) P(z)$$

This expression requires the analyst only to observe the idle (or unmanipulated) system and condition on such observations. If the set Z is chosen carefully we can calculate or estimate $P(y_0 \mid do x_0)$ from a partially observed idle system (ie. without conditioning on the full set of measurement variables). Lauritzen [30] and Dawid [12] have similar versions of this theorem.

One rather useful aspect of the theorem is that the conditions on the variables can be expressed graphically (that is, on the BN of the problem).

Example 1:

Consider the BN in Figure 17 and the manipulation $do X = x_0$.

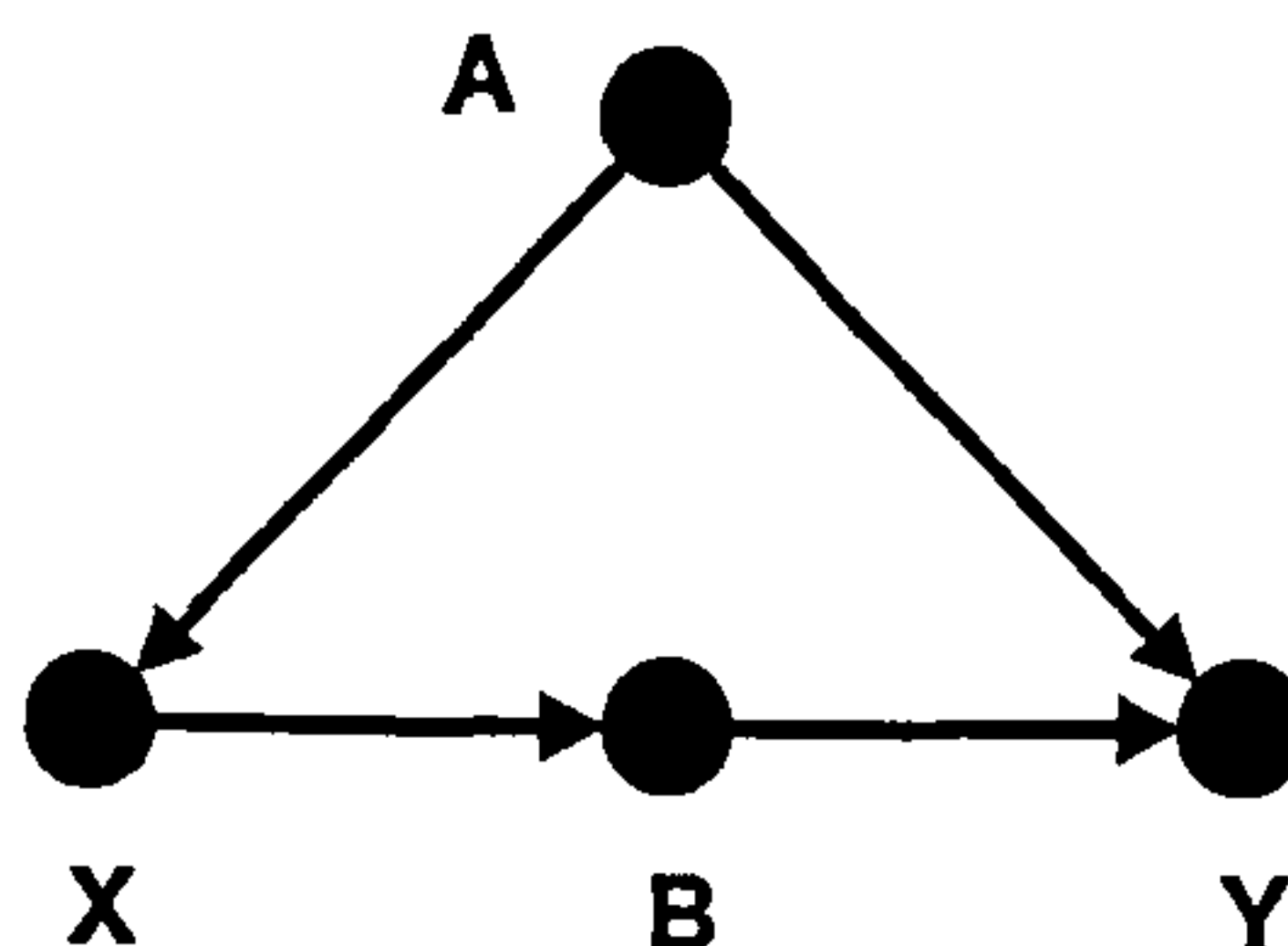


Figure 17: BN for Example 1

The manipulated BN here would be as in Figure 17, but with the directed edge from A to X removed. The manipulated probability expression here is:

$$\pi(y_0 \mid do\ x_0) = \sum_{a,b} \pi(a) \pi(b \mid x_0) \pi(y_0 \mid a, b)$$

This is an expression that can be evaluated on the idle system, but suppose it is impossible to estimate $\pi(b \mid x_0)$ or $\pi(y_0 \mid ab)$ for some values of a, b . The Back Door expression for this manipulation is:

$$\pi(y_0 \mid do\ x_0) = \sum_a \pi(a) \pi(y_0 \mid a, x_0)$$

a simpler expression for which we do not need to know about the values taken by the variable B .

Now the CEG is a much more flexible graph than the BN, so if we can produce a Back Door Theorem for CEGs, it is likely to be valid for a much larger collection of types of manipulation than are possible with a BN. Moreover, our theorem will refer to manipulations to events rather than manipulations of variables, which is much more consistent with our experience of what a manipulation really involves – in our Machine example we have not even defined any variables, we have simply manipulated to a specified event. As with the BN-version of the theorem, we should be able to reduce the complexity of the general manipulated-probability expression, as well as possibly allowing us to sidestep identifiability problems associated with it.

I start by expressing a Back Door Theorem for CEGs in its most general form. I then produce a Back Door Theorem for what I term *Singular* manipulations, which allows us to compare our theorem directly with that for BNs. I then show that we can express at least one of our conditions for this type of manipulation graphically (that is, on the CEG of the problem).

As always, my work here is based on the collection of atomic $w_0 \rightarrow w_\infty$ paths of the CEG, whereas Smith and Riccomagno [45] are more interested in the idea of *cuts* (section 3.5). As we are both (at least initially) looking at a Back Door Theorem for manipulations that are directly analogous to those that one can enact on a BN, there will be some superficial similarities.

The conditioned CEG C_Λ is a mathematically elegant structure, but as already noted, for practical (particularly computer-based) probability propagation, the undirected edges

are irrelevant and the combining of positions in the final step of the algorithm actually counterproductive.

Similarly, reading through the example in section 5.4 and also Example 4 from section 4.9 we can see that there could be significant advantages in looking at a manipulated or conditioned graph that retains as much as possible of the structure of the original graph C .

We therefore now give names to two graphs closely related to C_Λ and $C_{do\Lambda}$ which retain more of the original structure of C . We also formally define $C_{do\Lambda}$ and Singular manipulations in this section.

The *Derived Graph conditioned on Λ* (D_Λ) is the graph derived from C by following the first three steps of the algorithm from section 4.7. Clearly D_Λ is $G_\Lambda(C)$ with added probability labels.

Definition 29: For a CEG $C(W, E_d, E_u)$, and event $\Lambda = \bigcup_{i \in I} \lambda_i$, let $D_\Lambda(V, E)$ be the subgraph of C with:

- (a) $V(D_\Lambda) \subset W(C)$ contains exactly those positions which lie on a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
- (b) $E(D_\Lambda) \subset E_d(C)$ contains exactly those edges which form part of a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
[Note that $E(D_\Lambda) \cap E_u(C) = \phi$]
- (c) For $w_1, w_2 \in V(D_\Lambda)$ and $e(w_1, w_2) \in E(D_\Lambda)$, the edge $e(w_1, w_2)$ has probability:

$$\hat{\pi}_e(w_2 \mid w_1) = \frac{\sum_{i \in I} \pi(\lambda_i, \Lambda(e(w_1, w_2)))}{\sum_{i \in I} \pi(\lambda_i, \Lambda(w_1))}$$

where $\hat{\pi}$ indicates a probability in D_Λ and π a probability in C .

Note that (i) D_Λ has no undirected edges or colours, (ii) we continue to call vertices in D_Λ positions.

The *Derived Graph manipulated to Λ* ($D_{do\Lambda}$) is the graph derived from C by following the first two steps of the algorithm from section 4.7 (which gives us $G_\Lambda(C)$), and then adding edge-probabilities as implied by the manipulation $do \Lambda$.

Definition 30: For a CEG $C(W, E_d, E_u)$, and event $\Lambda = \bigcup_{i \in I} \lambda_i$, let $D_{do\Lambda}(V, E)$ be the subgraph of C with:

- (a) $V(D_{do\Lambda}) \subset W(C)$ contains exactly those positions which lie on a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
- (b) $E(D_{do\Lambda}) \subset E_d(C)$ contains exactly those edges which form part of a path $\lambda_i \in \{\lambda_i\}_{i \in I}$
[Note that $E(D_\Lambda) \cap E_u(C) = \phi$]

- (c) For $w_1, w_2 \in V(D_{do\Lambda})$ and $e(w_1, w_2) \in E(D_{do\Lambda})$, the edge $e(w_1, w_2)$ has probability uniquely assigned by the definition of the manipulation $do \Lambda$

Note that (i) $D_{do\Lambda}$ has no undirected edges or colours, (ii) we continue to call vertices in $D_{do\Lambda}$ positions.

Definition 31: $C_{do\Lambda}$ is the unique CEG derived from the graph $D_{do\Lambda}$ (see Definitions 9 and 13).

Note also that the onus is now to ensure that any manipulation $do \Lambda$ is unambiguously defined.

It is worth stressing two key properties of these graphs:

- All positions and edges in D_Λ and $D_{do\Lambda}$ exist in C
- There is a 1:1 correspondence between paths in D_Λ and in C_Λ , and between paths in $D_{do\Lambda}$ and in $C_{do\Lambda}$

We will indicate probabilities on D_Λ by π_Λ , and on $D_{do\Lambda}$ by $\pi_{do\Lambda}$.

Definition 32: A manipulation $do \Lambda$ of a CEG C is called a *Singular* manipulation if there exist sets $\{w\} \subset W(C)$ and $E(\Lambda) \subset E_d(C)$, such that:

- every $w_0 \rightarrow w_\infty$ path in C passes through precisely one $w \in \{w\}$,
- for each $w \in \{w\}$, there exists precisely one edge $e(w, w')$ from the set of edges emanating from w in $E_d(C)$, which is an element of $E(\Lambda)$,
- Λ is the union of precisely those $w_0 \rightarrow w_\infty$ paths that pass through an edge $e(w, w') \in E(\Lambda)$,
- all edge-probabilities in $D_{do\Lambda}$ are equal to their corresponding edge-probabilities in C , except that $\pi_{do\Lambda}.e(w' \mid w) = 1$ for $w \in \{w\}$ and $e(w, w') \in E(\Lambda)$.

Essentially, a Singular manipulation is one where every $w_0 \rightarrow w_\infty$ path passes through one of a collection of positions, and the manipulation imposes a probability of 1 on one edge emanating from each of these positions.

5.5.2 The general form of the theorem

Consider a manipulation $do \Lambda_x$. Suppose we wish to find the probability of (observing) an event Λ_y given that the manipulation $do \Lambda_x$ has been enacted – that is we wish to produce an expression for $\pi(\Lambda_y \mid do \Lambda_x)$. This is equal to the probability of the event Λ_y on the CEG $C_{do\Lambda_x}$, which is the sum of the probabilities of the $w_0 \rightarrow w_\infty$ paths in $C_{do\Lambda_x}$, which

are consistent with the event Λ_y . From the previous section we know that there is a 1:1 correspondence between paths in $C_{do\Lambda_x}$ and $D_{do\Lambda_x}$, and that every edge in $D_{do\Lambda_x}$ has the same probability as the corresponding edge in $C_{do\Lambda_x}$. Therefore we know that:

$$\pi(\Lambda_y \mid do \Lambda_x) = \pi_{do\Lambda_x}(\Lambda_y)$$

Consider a partition of the atomic events ($w_0 \rightarrow w_\infty$ paths in C): $\{\Lambda_z\}$. Then:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y) &= \pi_{do\Lambda_x}\left(\bigcup_z \Lambda_z, \Lambda_y\right) \\ &= \sum_z \pi_{do\Lambda_x}(\Lambda_z, \Lambda_y) \end{aligned}$$

since the events $\{\Lambda_z\}$ and hence the events $\{\Lambda_z, \Lambda_y\}$ are disjoint

$$= \sum_z \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda_z) \pi_{do\Lambda_x}(\Lambda_z)$$

Note that if a particular Λ_z were inconsistent with Λ_x then the paths comprising this Λ_z would not appear in $D_{do\Lambda_x}$ and hence $\pi_{do\Lambda_x}(\Lambda_z, \Lambda_y) = 0$ for this Λ_z .

The partition $\{\Lambda_z\}$ forms a *Back Door partition* of the atomic events if:

$$(A) \quad \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$$

$$(B) \quad \pi_{do\Lambda_x}(\Lambda_z) = \pi(\Lambda_z) \quad \text{for all } \Lambda_z \in \{\Lambda_z\}$$

If these conditions are satisfied then:

$$\pi(\Lambda_y \mid do \Lambda_x) = \pi_{do\Lambda_x}(\Lambda_y) = \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \pi(\Lambda_z)$$

This expression is clearly analogous in some way to Pearl's expression quoted in section 5.5.1. Exactly how the analogy works will depend on how we define the events Λ_x, Λ_y and $\{\Lambda_z\}$. As with the BN version, if we choose $\{\Lambda_z\}$ carefully, we can calculate or estimate $\pi(\Lambda_y \mid do \Lambda_x)$ from a partially observed idle system.

5.5.3 Singular manipulations

The flexibility of the CEG allows us (within the constraints that Λ_x etc. are of the form $\bigcup \lambda_i$) to define our events Λ_x, Λ_y and $\{\Lambda_z\}$ in many ways (for example, we could insist that Λ_x be of the form $\bigcup \Lambda(w)$), and hence conditions (A) and (B) can be satisfied in many ways. We are going to concentrate for the moment on the class of Singular manipulations. Our reason for doing so is that we wish initially to show how our result is directly analogous to that for BNs; and the class of interventions that can satisfactorily be analysed using a BN is a subset of the class of Singular manipulations. We will

also, for the present, concentrate on the subclass of Singular manipulations consisting of interventions of the form $do(X = x_0)$ for some criterion variable X . It should be stressed that this is for ease of exposition, and also to emphasise the direct analogy with BNs; and that the theoretical apparatus here presented can easily be adapted for general Singular manipulations. Again, we note that the class of interventions that can satisfactorily be analysed using a BN is a subset of this class.

As we work through the following sections, it should become apparent that our theoretical apparatus, as well as being ideal for use with asymmetric models, is also more flexible than Pearl's when used with symmetric models as it considers a Back Door partition of events rather than a Back Door set of variables.

Suppose every $w_0 \rightarrow w_\infty$ path in C passes through one of a set of positions $\{w_X\}$, each of which has the criterion variable X , and suppose that the event Λ_x simply assigns a probability of 1 to the edges leaving the positions $\{w_X\}$ which are labelled x_0 . Then:

$$\Lambda_x = \bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w'_X))]$$

where $e(w_X, w'_X)$ indicates the edge leaving w_X labelled x_0 .

Note also that we can write the manipulation $do \Lambda_x$ as $do(X = x_0)$ or simply $do x_0$.

It is worthwhile at this point looking at $D_{do\Lambda_x}$ for a manipulation of this sort. Using Definition 30, we can see that we create $D_{do\Lambda_x}$ as follows:

- (1) Remove all undirected edges from C .
- (2) Assign a probability of 1 to edges leaving positions $\{w_X\}$ which are labelled x_0 . Prune all positions and edges which are not consistent with the event Λ_x .
- (3) Retain the original probabilities on all other remaining edges.

Let us also briefly consider D_{Λ_x} , the Derived Graph conditioned on the event Λ_x , for Λ_x as described above (it might have been noted that condition (A) from section 5.2.2 could be written as $\pi_{do\Lambda_x}(\Lambda_y | \Lambda_z) = \pi_{\Lambda_x}(\Lambda_y | \Lambda_z)$). From section 5.5.1 it is clear that the topology of D_{Λ_x} is identical to that of $D_{do\Lambda_x}$ and that there is therefore a 1:1 correspondence between $w_0 \rightarrow w_\infty$ paths in the two graphs.

Consider now positions w_1 and w_2 in D_{Λ_x} such that $w_X \prec w_1 \prec w_2$ for some $w_X \in \{w_X\}$ (note that we could have $w_1 = w'_X$). Then we get:

$$\pi_{\Lambda_x \cdot e}(w_2 | w_1) = \frac{\pi(\Lambda_x, \Lambda(e(w_1, w_2)))}{\pi(\Lambda_x, \Lambda(w_1))}$$

(contracting the edge-probability expression in part (c) of Definition 29)

$$\begin{aligned}
&= \frac{\pi(\bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w'_X))], \Lambda(e(w_1, w_2)))}{\pi(\bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w'_X))], \Lambda(w_1))} \\
&= \frac{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(e(w_1, w_2)))}{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))} \\
&\text{since the events } \{\Lambda(w_X), \Lambda(e(w_X, w'_X))\} \text{ are disjoint} \\
&= \frac{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1), \Lambda(e(w_1, w_2)))}{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))} \\
&\text{since } \Lambda(e(w_1, w_2)) \subset \Lambda(w_1) \\
&= \frac{\sum_{w_X} \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1)) \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))}{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))} \\
&= \frac{\sum_{w_X} \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1)) \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))}{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))} \\
&\text{using one of class of results spawned by Proposition 5 in section 3.6} \\
&= \frac{\pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1)) \sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))}{\sum_{w_X} \pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w_1))} \\
&= \pi(\Lambda(e(w_1, w_2)) \mid \Lambda(w_1)) \\
&= \pi_e(w_2 \mid w_1)
\end{aligned}$$

So any edge downstream of any $w_X \in \{w_X\}$ in D_{Λ_x} retains its original probability, and consequently any path-segment starting downstream of any w_X in D_{Λ_x} also retains its original probability. Likewise, we know that all edges downstream of w_X in $D_{do\Lambda_x}$ retain their original probabilities, and consequently any path-segments starting downstream of any w_X in $D_{do\Lambda_x}$ also retain their original probabilities.

As there is a 1:1 correspondence between $w_0 \rightarrow w_\infty$ paths in D_{Λ_x} and $D_{do\Lambda_x}$, we can therefore state that any path-segments starting downstream of w_X that exist in D_{Λ_x} , exist in $D_{do\Lambda_x}$, and vice versa, and have the same probabilities. This will be useful a little later on!

5.5.4 The theorem for Singular manipulations

Suppose now that every $w_0 \rightarrow w_\infty$ path in C passes through one of a set of positions $\{w_Y\}$ (downstream of the set $\{w_X\}$), each of which has the criterion variable Y , and suppose that the event Λ_y simply assigns a probability of 1 to the edges leaving the positions $\{w_Y\}$ which are labelled y_0 . Then:

$$\Lambda_y = \bigcup_{w_Y} [\Lambda(w_Y) \cap \Lambda(e(w_Y, w'_Y))]$$

where $e(w_Y, w'_Y)$ indicates the edge leaving w_Y labelled y_0 .

Note also that we can write $\pi(\Lambda_y \mid do \Lambda_x) = \pi(y_0 \mid do x_0)$.

It is very useful at this point to look at the conditions for Pearl's Back Door Theorem on BNs. He has two conditions, both of which can be re-expressed as conditional independence statements: Pearl's condition that Z (the Back Door *blocking set*) must *block* all *Back Door* paths from X to Y can be expressed as the conditional independence statement:

$$Y \perp\!\!\!\perp Q(X) \mid (X, Z)$$

where $Q(X)$ indicates the variable-parents of X . In Example 1, A is both $Q(X)$ and Z (see section 5.5.8) – it *blocks* paths from X to Y which have a directed edge *into* X .

Pearl's condition that Z must contain no descendants of X can be expressed as the conditional independence statement:

$$Z \perp\!\!\!\perp X \mid Q(X)$$

In Example 1, A is an ancestor of X .

In our discussion of singular manipulations, we have already replaced $X = x$ by $\Lambda_x = \bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w'_X))]$ and $Y = y$ by $\Lambda_y = \bigcup_{w_Y} [\Lambda(w_Y) \cap \Lambda(e(w_Y, w'_Y))]$.

We now replace $Q(X) = q(x)$ by $\Lambda(w_X)$, and $Z = z$ by Λ_z .

If we now substitute these into the conditional independence statements above, we get, for $Z \perp\!\!\!\perp X \mid Q(X)$:

$$\pi(\Lambda_z \mid \Lambda(w_X), \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))]) = \pi(\Lambda_z \mid \Lambda(w_X))$$

But

$$\Lambda(w_X) \cap [\bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w'_X))]] = \Lambda(w_X) \cap \Lambda(e(w_X, w'_X))$$

so we get:

$$\pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) = \pi(\Lambda_z \mid \Lambda(w_X)) \quad (5.5.1)$$

Applying our analogous expressions to the condition $Y \perp\!\!\!\perp Q(X) \mid (X, Z)$, we get:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda(w_X), \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))], \Lambda_z) &= \pi(\Lambda_y \mid \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))], \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \end{aligned}$$

But

$$\pi(\Lambda_y \mid \Lambda(w_X), \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))], \Lambda_z) = \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda_z)$$

using the same argument as above, so we get a 2nd condition:

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) &= \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)})), \Lambda_z)\end{aligned}\tag{5.5.2}$$

Proposition 13:

With Λ_x, Λ_y defined as above, and $\{\Lambda_z\}$ a partition of the atomic events, then $\{\Lambda_z\}$ is a Back Door partition if conditions (5.5.1) and (5.5.2) hold for all $\Lambda_z \in \{\Lambda_z\}, w_X \in \{w_X\}$.

Note that these conditions are on C , not $D_{do\Lambda_x}$ or $C_{do\Lambda_x}$. They can therefore, like Pearl's two conditions, be checked on an unmanipulated graph (a representation of the idle system).

Proof:

$$\pi_{do\Lambda_x}(\Lambda_y) = \sum_{w_X} \pi_{do\Lambda_x}(\Lambda(w_X), \Lambda_y)$$

since the events $\{\Lambda(w_X)\}$ form a partition of the atomic events

$$\begin{aligned}&= \sum_{w_X} \pi_{do\Lambda_x}(\Lambda(w_X)) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w_X)) \\ &= \sum_{w_X} \pi(\Lambda(w_X)) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w_X))\end{aligned}$$

since every w_X lies upstream of our manipulation

$$= \sum_{w_X} \pi(\Lambda(w_X)) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w'_X))$$

since $\Lambda(w_X) = \Lambda(e(w_X, w'_X)) \subset \Lambda(w'_X)$ in $D_{do\Lambda_x}$

The form we have specified for Λ_y , and the fact that $w_X \prec w'_X \prec w_Y$ for some w_Y in $D_{do\Lambda_x}$ allows us to use one of the class of results spawned by Proposition 5 (section 3.6) to give:

$$\pi_{do\Lambda_x}(\Lambda_y) = \sum_{w_X} \pi(\Lambda(w_X)) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w'_X))$$

From the definition of our manipulation, any edge that lies on a $w'_X \rightarrow w_\infty$ path in C remains in $D_{do\Lambda_x}$, and retains its original probability. Hence any path-segment in C which starts at w'_X remains in $D_{do\Lambda_x}$, and retains its original probability. There are no path-segments starting at w'_X in $D_{do\Lambda_x}$ which do not correspond to a path-segment in C . Hence any specified set of path-segments which start at w'_X in $D_{do\Lambda_x}$ has a corresponding set of path-segments (starting at w'_X) in C , and has the same probability as this set. Given

the form specified for Λ_y , $\pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w'_X))$ is the probability of a set of path-segments starting at w'_X in $D_{do\Lambda_x}$ (see Proposition 4 in section 3.6). Hence

$$\pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w'_X)) = \pi(\Lambda_y \mid \Lambda(w'_X))$$

and:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y) &= \sum_{w_X} \pi(\Lambda(w_X)) \pi(\Lambda_y \mid \Lambda(w'_X)) \\ &= \sum_{w_X} \pi(\Lambda(w_X)) \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda(w'_X)) \end{aligned}$$

using the form of Λ_y , the fact that $w_X \prec w'_X \prec w_Y$ for some w_Y in C , and one of the class of results spawned by Proposition 5

$$= \sum_{w_X} \pi(\Lambda(w_X)) \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)))$$

since $\Lambda(e(w_X, w'_X)) \subset \Lambda(w'_X)$ in C

$$= \sum_{w_X} \pi(\Lambda(w_X)) \sum_z \pi(\Lambda_z, \Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)))$$

since $\{\Lambda_z\}$ form a partition of the atomic events

$$= \sum_{w_X} \pi(\Lambda(w_X)) \sum_z \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda_z) \pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)))$$

But:

$$\pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) = \pi(\Lambda_z \mid \Lambda(w_X))$$

from (5.5.1), and

$$\pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$$

from (5.5.2), giving:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y) &= \sum_{w_X} \pi(\Lambda(w_X)) \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \pi(\Lambda_z \mid \Lambda(w_X)) \\ &= \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \sum_{w_X} \pi(\Lambda_z \mid \Lambda(w_X)) \pi(\Lambda(w_X)) \\ &= \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \pi(\Lambda_z) \end{aligned}$$

□

5.5.5 A graphical condition

In section 5.5.1 I noted that Pearl's conditions for his Back Door Theorem could be expressed graphically. Are there restrictions we can impose on the topology of our CEG C which mean that either or both of conditions (5.5.1) and (5.5.2) are satisfied?

Suppose each element of the partition $\{\Lambda_z\}$ is of the form:

$$\Lambda_z = \bigcup_{i \in I} \Lambda(w_z^i)$$

where (i) $\Lambda(w_z^i) \cap \Lambda(w_z^j) = \phi$ for $i, j \in I$, $i \neq j$, (ii) only w_z^1 exists in $D_{do\Lambda_x}$.

The complete set of w_z (over all Λ_z) here forms a W -cut (section 3.5) across the CEG. Let this set $\{w_z\}$ be downstream of the set $\{w_X\}$ and upstream of the set of edges labelled y_0 .

Then we have:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) \\ = \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda(w'_{X(1)}), \Lambda_z) \end{aligned}$$

since $\Lambda(e(w_{X(1)}, w'_{X(1)})) \subset \Lambda(w'_{X(1)})$ in C

$$\begin{aligned} &= \frac{\pi(\Lambda_z, \Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda(w'_{X(1)}))}{\pi(\Lambda_z \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda(w'_{X(1)}))} \\ &= \frac{\pi(\Lambda_z, \Lambda_y \mid \Lambda(w'_{X(1)}))}{\pi(\Lambda_z \mid \Lambda(w'_{X(1)}))} \end{aligned}$$

We can do this because $\Lambda_z = \bigcup \Lambda(w_z)$, $\Lambda_y = \bigcup [\Lambda(w_Y) \cap \Lambda(e(w_Y, w'_Y))]$, and $\{w_z\}, \{w_Y\}$ are downstream of $\{w_X\}$, allowing us to employ one of the class of results spawned by Proposition 5. Hence we get:

$$\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) = \pi(\Lambda_y \mid \Lambda(w'_{X(1)}), \Lambda_z) \quad (5.5.3)$$

Similarly:

$$\pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)})), \Lambda_z) = \pi(\Lambda_y \mid \Lambda(w'_{X(2)}), \Lambda_z) \quad (5.5.4)$$

But any path-segment in C starting at $w'_{X(1)}$ or $w'_{X(2)}$ remains in $D_{do\Lambda_x}$, and we know that $\{w_z^i\}_{i \geq 2}$ do not exist in $D_{do\Lambda_x}$, so there are no path-segments joining $w'_{X(1)}$ or $w'_{X(2)}$ to w_z^i (for $i \geq 2$) in $D_{do\Lambda_x}$, and hence no path-segments joining $w'_{X(1)}$ or $w'_{X(2)}$ to w_z^i (for $i \geq 2$) in C . Hence we must have:

$$\Lambda(w'_{X(1)}) \cap \Lambda(w_z^i) = \Lambda(w'_{X(2)}) \cap \Lambda(w_z^i) = \phi \quad \text{for } i \geq 2$$

so:

$$\Lambda(w'_{X(1)}) \cap \Lambda_z = \Lambda(w'_{X(1)}) \cap \Lambda(w_z^1)$$

$$\Lambda(w'_{X(2)}) \cap \Lambda_z = \Lambda(w'_{X(2)}) \cap \Lambda(w_z^1)$$

and (5.5.3) becomes:

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w'_{X(1)}), \Lambda(w_z^1)) \\ &= \pi(\Lambda_y \mid \Lambda(w_z^1))\end{aligned}$$

using the specified form of Λ_y , the fact that $\{w_z\}$ are downstream of $\{w_X\}$, and one of the results spawned by Proposition 5

Similarly (5.5.4) becomes:

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)})), \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w'_{X(2)}), \Lambda(w_z^1)) \\ &= \pi(\Lambda_y \mid \Lambda(w_z^1))\end{aligned}$$

and hence:

$$\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) = \pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)})), \Lambda_z)$$

as required for condition (5.5.2).

So, if we choose each Λ_z to be of the form $\Lambda_z = \bigcup_{i \in I} \Lambda(w_z^i)$, where only w_z^1 exists in $D_{do\Lambda_z}$, then this is sufficient for condition (5.5.2) to be satisfied.

Note that our argument above depends on Λ_z having this form, as $\pi(\Lambda_y \mid \Lambda(w'_{X(1)}), \Lambda_z)$ is not in general equal to $\pi(\Lambda_y \mid \Lambda(w'_{X(2)}), \Lambda_z)$ for general Λ_z , nor for Λ_z of the form $\Lambda_z = \bigcup \Lambda(w_z)$ unless $\Lambda(w'_{X(1)}) \cap \Lambda(w_z^i) = \Lambda(w'_{X(2)}) \cap \Lambda(w_z^i) = \phi$ for all w_z^i except for one (w_z^1 say). In fact, if $\Lambda_z = \bigcup \Lambda(w_z)$, then $\pi(\Lambda_y \mid \Lambda(w'_X), \Lambda_z)$ is not in general even the probability of a set of path-segments, as can be seen in Figure 18, where $\Lambda_z = \bigcup_i \Lambda(w_z^i)$, and the vertex w_Y stands in for the set of vertices and edges that appear in Λ_y .

$$\begin{aligned}\pi(\Lambda(w_Y) \mid \Lambda(w'_X), \Lambda_z) &= \pi(\Lambda(w_Y) \mid \Lambda(w'_X), \bigcup_i \Lambda(w_z^i)) \\ &= \frac{\sum_i \pi(\Lambda(w'_X), \Lambda(w_z^i), \Lambda(w_Y))}{\sum_i \pi(\Lambda(w'_X), \Lambda(w_z^i))}\end{aligned}$$

since $\Lambda(w_z^i) \cap \Lambda(w_z^j) = \phi$ for $i \neq j$

$$\begin{aligned}&= \frac{\sum_i \pi(\Lambda(w_Y) \mid \Lambda(w'_X), \Lambda(w_z^i)) \pi(\Lambda(w'_X), \Lambda(w_z^i))}{\sum_i \pi(\Lambda(w'_X), \Lambda(w_z^i))} \\ &= \frac{\sum_i \pi(\Lambda(w_Y) \mid \Lambda(w_z^i)) \pi(\Lambda(w'_X), \Lambda(w_z^i))}{\sum_i \pi(\Lambda(w'_X), \Lambda(w_z^i))}\end{aligned}$$

using Proposition 5

$$= \frac{\sum_i \pi_i \pi(\Lambda(w'_X), \Lambda(w_z^i))}{\sum_i \pi(\Lambda(w'_X), \Lambda(w_z^i))}$$

is not in general equal to $\sum_i \pi_i$.

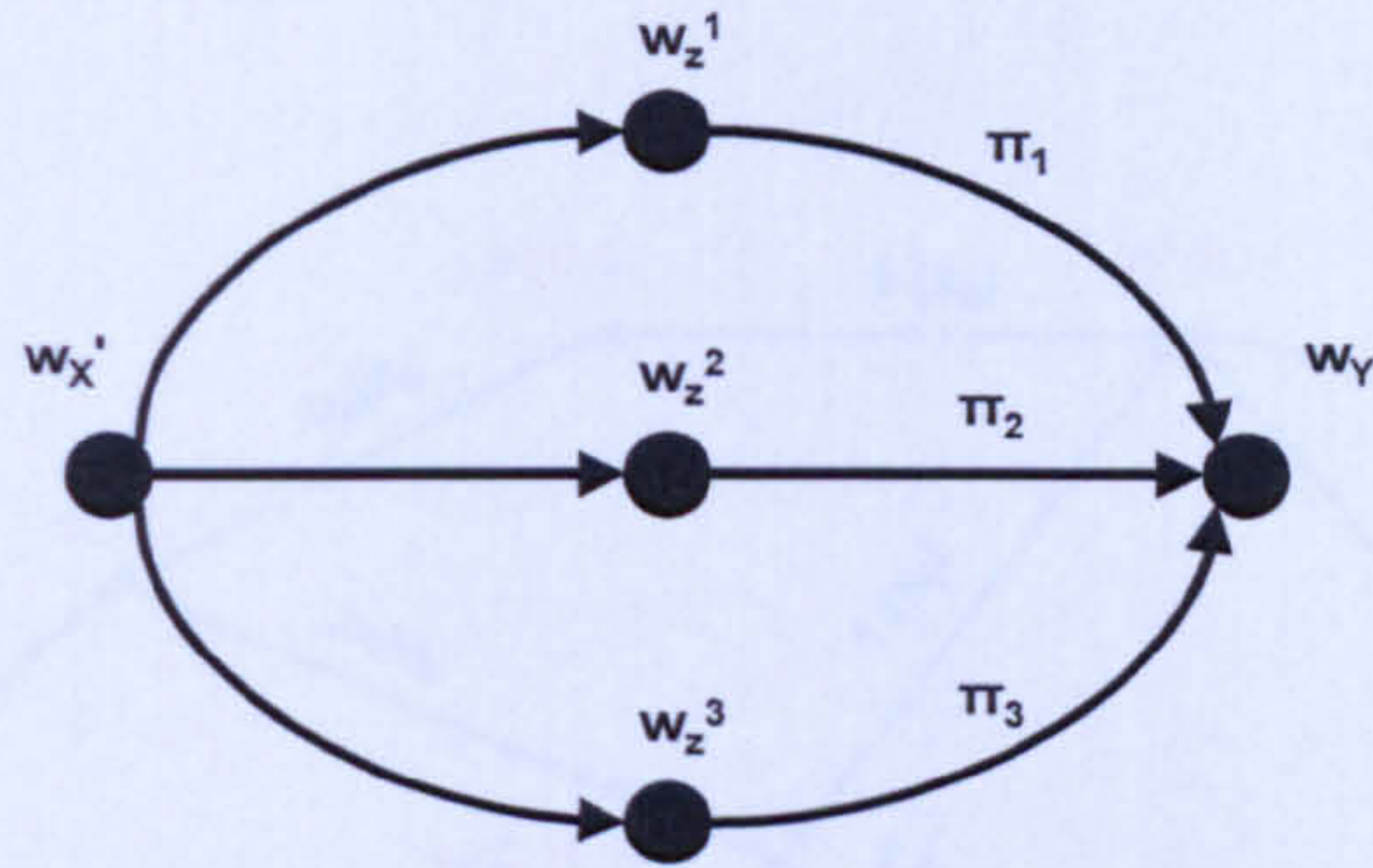


Figure 18: CEG fragment

So, condition (5.5.1) and this new graphical condition on the form of Λ_z are sufficient for $\{\Lambda_z\}$ to be a Back Door partition of the atomic events.

5.5.6 Examples

In this section we illustrate how our conditions work in practice. I start with a problem which can be represented by both a CEG and a BN, and consider a manipulation which can be defined on a BN:

Example 2:

Consider the BN and corresponding CEG in Figure 19:

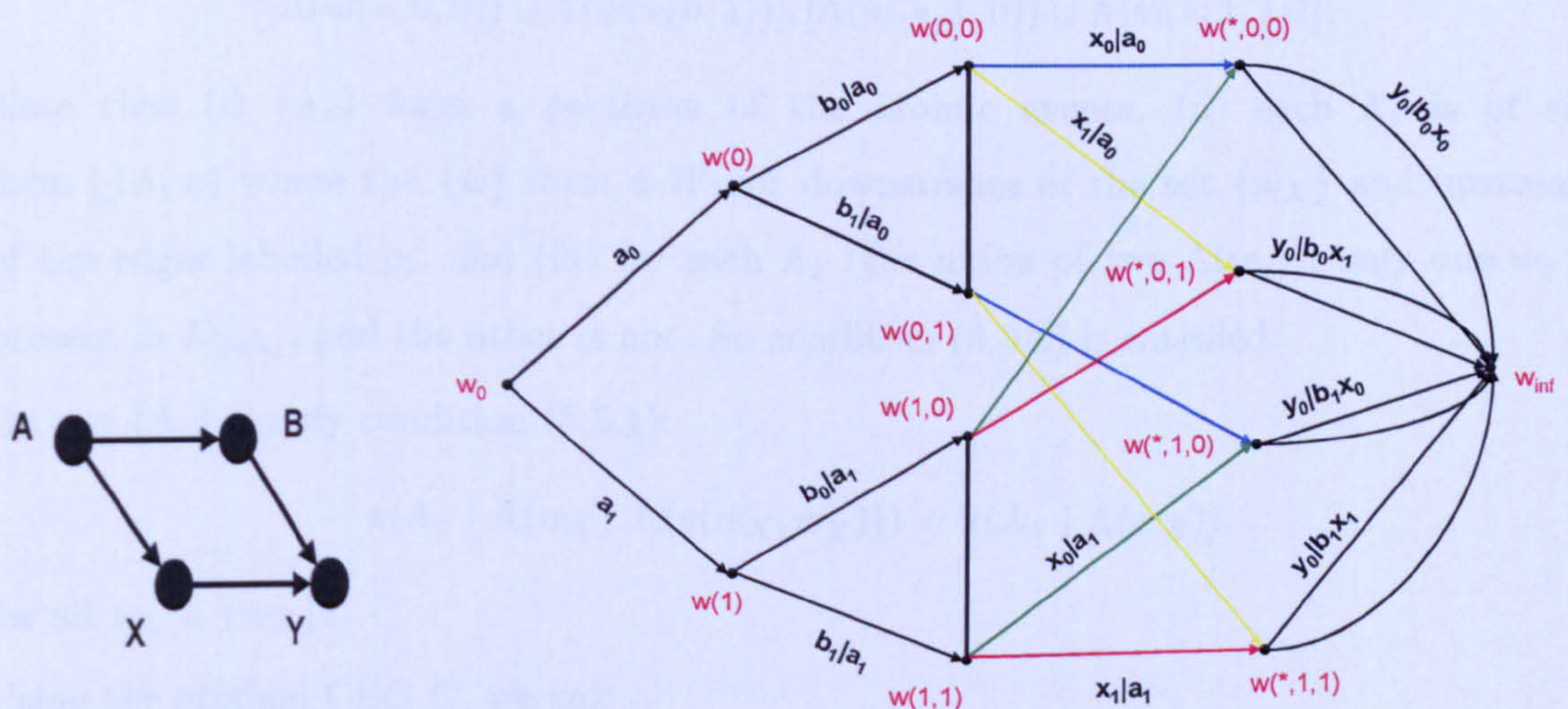


Figure 19: Bayesian Network and CEG for Example 2

Let us consider the manipulation $do \Lambda_x = do (X = x_0)$, the event $\Lambda_y = (Y = y_0)$, and the expression $\pi(\Lambda_y \mid do \Lambda_x) = \pi(y_0 \mid do x_0)$. Our Derived Graph manipulated to the

event $\Lambda_x (D_{do\Lambda_x})$ is given in Figure 20.

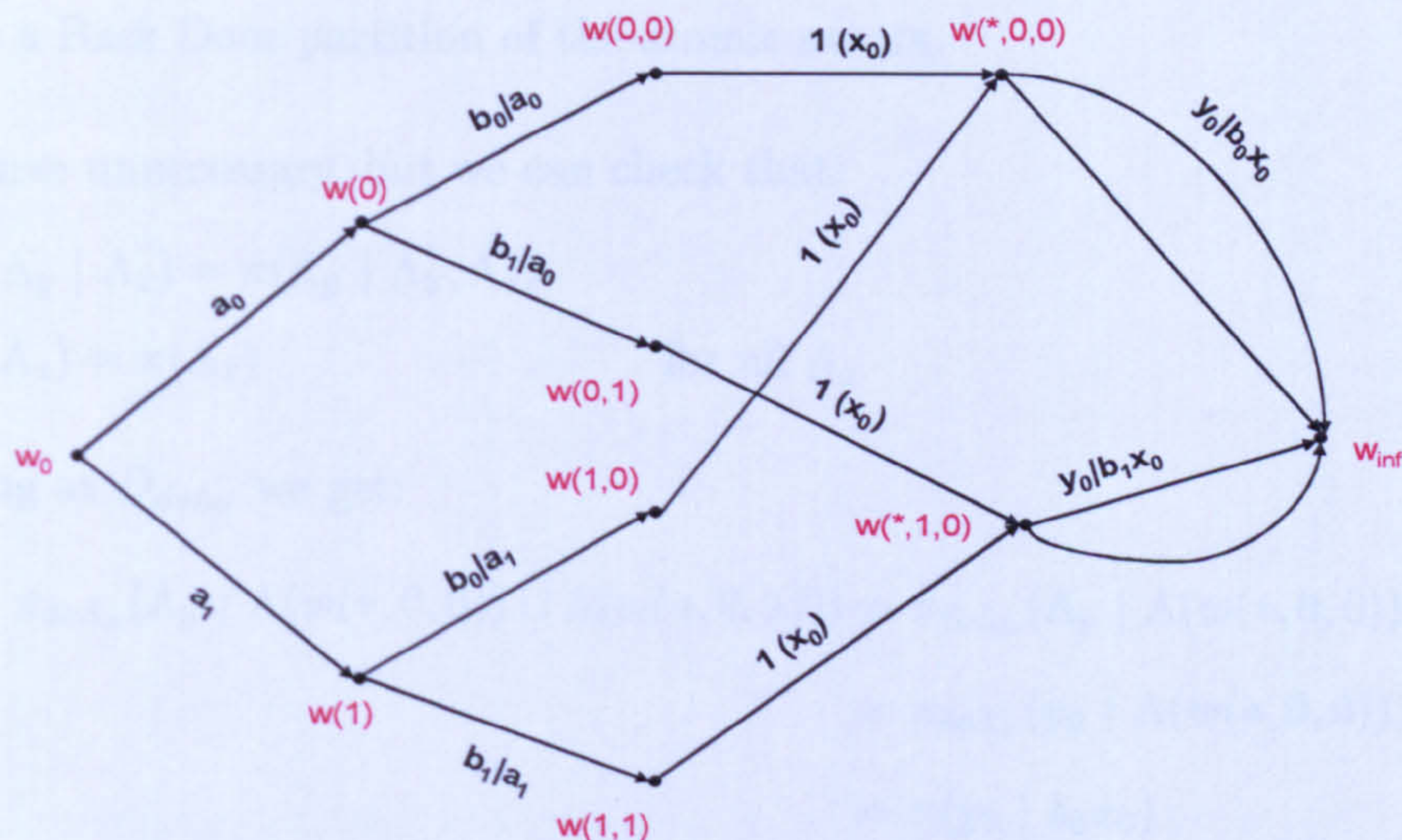


Figure 20: Derived Graph $D_{do\Lambda_x}$ for Example 2

Then $\{w_X\}$ and $\{e(w_X, w'_X)\}$ are respectively:

$$\{w(0,0), w(0,1), w(1,0), w(1,1)\} \quad \text{and}$$

$$\{e(w(0,0), w(*,0,0)), e(w(0,1), w(*,1,0)), e(w(1,0), w(*,0,0)), e(w(1,1), w(*,1,0))\}$$

We choose our $\{\Lambda_z\}$ to be:

$$\{[\Lambda(w(*,0,0)) \cup \Lambda(w(*,0,1))], [\Lambda(w(*,1,0)) \cup \Lambda(w(*,1,1))]\}$$

Note that (i) $\{\Lambda_z\}$ form a partition of the atomic events, (ii) each Λ_z is of the form $\bigcup \Lambda(w)$ where the $\{w\}$ form a W -cut downstream of the set $\{w_X\}$ and upstream of the edges labelled y_0 , and (iii) for each Λ_z (the union of two $\Lambda(w_z)$), only one w_z is present in $D_{do\Lambda_x}$, and the other is not. So condition (5.5.2) is satisfied.

Do our $\{\Lambda_z\}$ satisfy condition (5.5.1):

$$\pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) = \pi(\Lambda_z \mid \Lambda(w_X))$$

for all $w_X \in \{w_X\}$?

Using the original CEG C , we get:

$$\begin{aligned} & \pi(\Lambda(w(*,0,0)) \cup \Lambda(w(*,0,1)) \mid \Lambda(w(0,0)), \Lambda(e(w(0,0), w(*,0,0)))) \\ &= \pi(\Lambda(w(*,0,0)) \mid \Lambda(w(0,0)), \Lambda(e(w(0,0), w(*,0,0)))) = 1 \\ & \pi(\Lambda(w(*,0,0)) \cup \Lambda(w(*,0,1)) \mid \Lambda(w(0,0))) \\ &= \pi(x_0 \mid a_0) + \pi(x_1 \mid a_0) = 1 \checkmark \end{aligned}$$

Similarly for $\Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))$ and $w(1, 0)$, $\Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))$ and $w(0, 1)$, and $\Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))$ and $w(1, 1)$.

So $\{\Lambda_z\}$ is a Back Door partition of the atomic events.

It is of course unnecessary, but we can check that:

$$(A) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$$

$$(B) \pi_{do\Lambda_x}(\Lambda_z) = \pi(\Lambda_z) \quad \text{for all } \Lambda_z$$

(A) Looking at $D_{do\Lambda_x}$ we get:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))) &= \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(*, 0, 0))) \\ &= \pi_{do\Lambda_x}(y_0 \mid \Lambda(w(*, 0, 0))) \\ &= \pi(y_0 \mid b_0 x_0) \end{aligned}$$

Similarly

$$\pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))) = \pi(y_0 \mid b_1 x_0)$$

Looking at C we get:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda_x, \Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))) &= \pi(y_0 \mid x_0, \Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))) \\ &= \pi(y_0 \mid x_0, \Lambda(w(*, 0, 0))) \\ &= \pi(y_0 \mid \Lambda(w(*, 0, 0))) \\ &= \pi(y_0 \mid b_0, x_0) \checkmark \end{aligned}$$

Similarly

$$\pi(\Lambda_y \mid \Lambda_x, \Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))) = \pi(y_0 \mid b_1 x_0) \checkmark$$

(B) Looking at $D_{do\Lambda_x}$ we get:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))) &= \pi_{do\Lambda_x}(\Lambda(w(*, 0, 0))) \\ &= \pi(a_0) \times \pi(b_0 \mid a_0) \times 1 + \pi(a_1) \times \pi(b_0 \mid a_1) \times 1 \\ &= \pi(b_0) \end{aligned}$$

Similarly

$$\pi_{do\Lambda_x}(\Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))) = \pi(b_1)$$

Looking at C we get:

$$\begin{aligned} \pi(\Lambda(w(*, 0, 0)) \cup \Lambda(w(*, 0, 1))) &= \pi(a_0) \pi(b_0 \mid a_0) \pi(x_0 \mid a_0) + \pi(a_0) \pi(b_0 \mid a_0) \pi(x_1 \mid a_0) \\ &\quad + \pi(a_1) \pi(b_0 \mid a_1) \pi(x_0 \mid a_1) + \pi(a_1) \pi(b_0 \mid a_1) \pi(x_1 \mid a_1) \\ &= \pi(b_0) \checkmark \end{aligned}$$

Similarly

$$\pi(\Lambda(w(*, 1, 0)) \cup \Lambda(w(*, 1, 1))) = \pi(b_1) \checkmark$$

We can therefore write:

$$\begin{aligned} \pi(y_0 \mid do\ x_0) &= \pi(\Lambda_y \mid do\ \Lambda_x) \\ &= \pi_{do\ \Lambda_x}(\Lambda_y) \\ &= \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \pi(\Lambda_z) \\ &= \pi(y_0 \mid b_0, x_0) \pi(b_0) + \pi(y_0 \mid b_1, x_0) \pi(b_1) \\ &= \sum_b \pi(y_0 \mid b, x_0) \pi(b) \end{aligned}$$

which is of course (one of) the expression(s) which we would get using Pearl's Back Door Theorem on the BN in Figure 19.

Note that this method works because each Λ_z here is the union of the events associated with two positions, both of which contribute to the term $\pi(\Lambda_z)$ in our probability expression, but only **one** of which exists in $D_{do\ \Lambda_x}$ and hence contributes to the term $\pi_{do\ \Lambda_x}(\Lambda_y \mid \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$.

Example 3:

We now consider a model which cannot adequately be described by a BN. The CEG for this model is given in Figure 21:

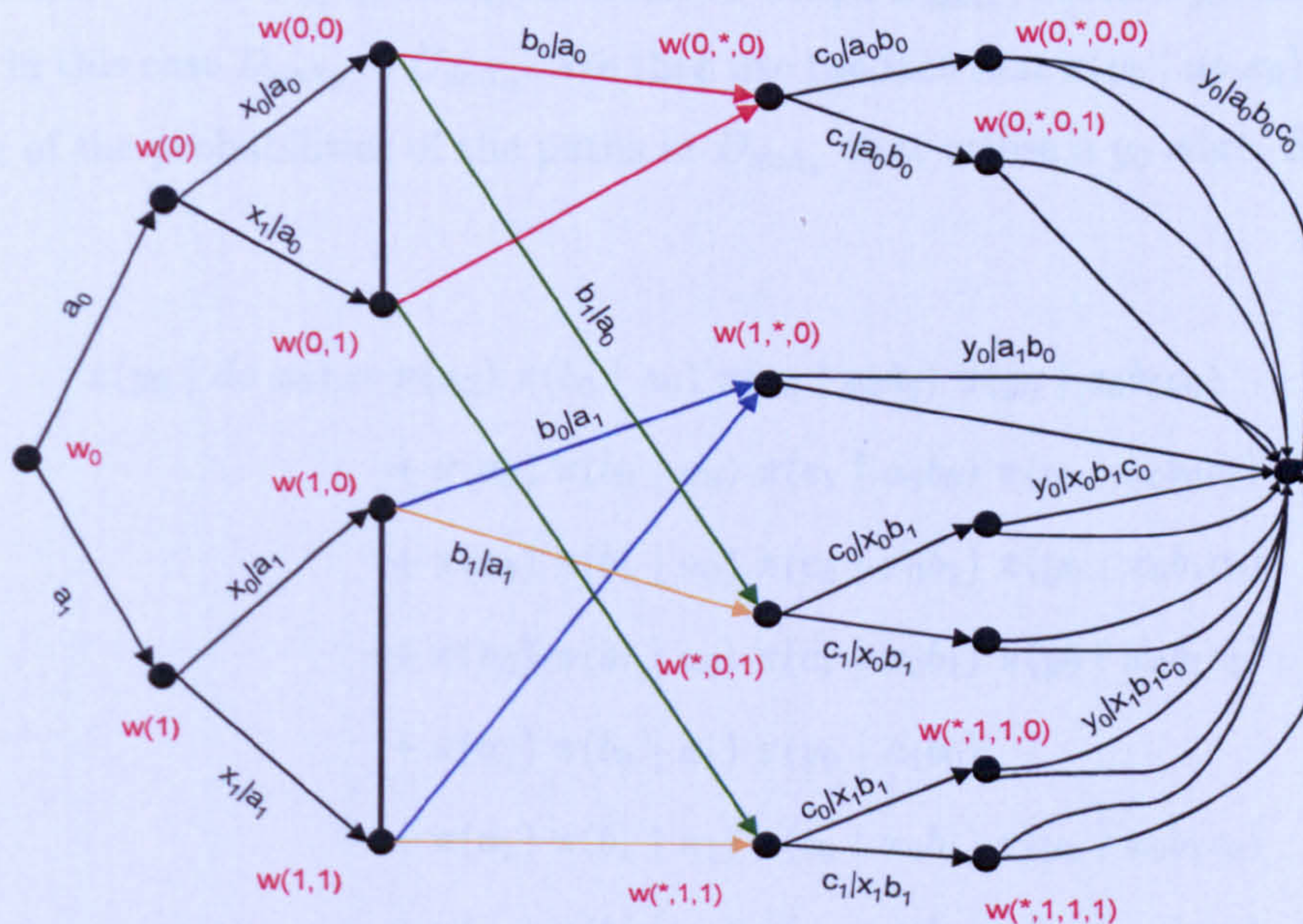


Figure 21: CEG for Example 3

The CEG has certain symmetries, for example in the edges leaving $w(0,0)$, $w(0,1)$, $w(1,0)$ and $w(1,1)$, and that every $w_0 \rightarrow w_\infty$ path utilises edges labelled a, x, b and y (but not c). It also has certain asymmetrical aspects, in that the positions have asymmetric sample space structures, but also in the fact that $w_0 \rightarrow w_\infty$ paths passing through $w(1,*,0)$ are *shorter* than the other $w_0 \rightarrow w_\infty$ paths. This mix of symmetry and asymmetry is not uncommon in real problems.

We are going to consider the Singular manipulation $do \Lambda_x = do (X = x_0)$, and wish to calculate the probability $\pi(\Lambda_y \mid do \Lambda_x) = \pi(y_0 \mid do x_0)$.

The CEG satisfies the conditions that every path passes through a position from $\{w_X\} = \{w(0), w(1)\}$ and a position from $\{w_Y\} = \{w(0,*,0,0), w(0,*,0,1), w(1,*,0), w(*,0,1,0), w(*,0,1,1), w(*,1,1,0), w(*,1,1,1)\}$. Also, every position in $\{w_X\}$ has an outgoing edge labelled x_0 , and every position in $\{w_Y\}$ has an outgoing edge labelled y_0 .

We have selected a set of positions $\{w_z\}$ lying between $\{w_X\}$ and the set of edges labelled y_0 , and this set comprises $\{w(0,*,0,0), w(0,*,0,1), w(1,*,0), w(*,0,1), w(*,1,1,0), w(*,1,1,1)\}$.

We now wish to calculate $\pi(y_0 \mid do x_0)$, and we do this first without recourse to our Back Door Theorem. We start by creating the Derived Graph $D_{do\Lambda_x}$, which is given in Figure 22. Note that in this case $D_{do\Lambda_x} = C_{do\Lambda_x}$. We then use the fact that $\pi(y_0 \mid do x_0) = \pi_{do\Lambda_x}(y_0)$ is the sum of the probabilities of the paths in $D_{do\Lambda_x}$ that utilise a y_0 edge. So:

$$\begin{aligned} \pi(y_0 \mid do x_0) &= \pi(a_0) \pi(b_0 \mid a_0) \pi(c_0 \mid a_0 b_0) \pi(y_0 \mid a_0 b_0 c_0) \\ &\quad + \pi(a_0) \pi(b_0 \mid a_0) \pi(c_1 \mid a_0 b_0) \pi(y_0 \mid a_0 b_0 c_1) \\ &\quad + \pi(a_0) \pi(b_1 \mid a_0) \pi(c_0 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_0) \\ &\quad + \pi(a_0) \pi(b_1 \mid a_0) \pi(c_1 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_1) \\ &\quad + \pi(a_1) \pi(b_0 \mid a_1) \pi(y_0 \mid a_1 b_0) \\ &\quad + \pi(a_1) \pi(b_1 \mid a_1) \pi(c_0 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_0) \\ &\quad + \pi(a_1) \pi(b_1 \mid a_1) \pi(c_1 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_1) \\ &= \pi(b_0) \pi(y_0 \mid b_0) + \pi(b_1) \pi(y_0 \mid x_0 b_1) \end{aligned}$$

$w_X = w(0) :$

$$\begin{aligned}
& \pi(\Lambda(w(0, *, 0, 0)) \mid \Lambda(w(0)), \Lambda(e(w(0), w(0, 0)))) \\
& \quad = \pi(b_0 \mid a_0) \pi(c_0 \mid a_0 b_0) \\
& \quad = \pi(b_0 c_0 \mid a_0) \\
& \pi(\Lambda(w(0, *, 0, 0)) \mid \Lambda(w(0))) = [\pi(x_0 \mid a_0) + \pi(x_1 \mid a_0)] \pi(b_0 \mid a_0) \pi(c_0 \mid a_0 b_0) \\
& \quad = \pi(b_0 c_0 \mid a_0) \checkmark
\end{aligned}$$

$w(0, *, 0, 1)$ follows exactly the same reasoning \checkmark

$w(1, *, 0)$ is not downstream of $w(0)$

$$\begin{aligned}
& \pi(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1)) \mid \Lambda(w(0)), \Lambda(e(w(0), w(0, 0)))) \\
& \quad = \pi(\Lambda(w(*, 0, 1)) \mid \Lambda(w(0)), \Lambda(e(w(0), w(0, 0)))) \\
& \quad = \pi(b_1 \mid a_0) \\
& \pi(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1)) \mid \Lambda(w(0))) \\
& \quad = \pi(x_0 \mid a_0) \pi(b_1 \mid a_0) \\
& \quad \quad + \pi(x_1 \mid a_0) \pi(b_1 \mid a_0) \pi(c_0 \mid x_1 b_1) \\
& \quad \quad + \pi(x_1 \mid a_0) \pi(b_1 \mid a_0) \pi(c_1 \mid x_1 b_1) \\
& \quad = \pi(b_1 \mid a_0) \checkmark
\end{aligned}$$

$w_X = w(1) :$

$w(0, *, 0, 0)$ and $w(0, *, 0, 1)$ are not downstream of $w(1)$

$$\begin{aligned}
& \pi(\Lambda(w(1, *, 0)) \mid \Lambda(w(1)), \Lambda(e(w(1), w(1, 0)))) = \pi(b_0 \mid a_1) \\
& \pi(\Lambda(w(1, *, 0)) \mid \Lambda(w(1))) = [\pi(x_0 \mid a_1) + \pi(x_1 \mid a_1)] \pi(b_0 \mid a_1) \\
& \quad = \pi(b_0 \mid a_1) \checkmark
\end{aligned}$$

$$\begin{aligned}
& \pi(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1)) \mid \Lambda(w(1)), \Lambda(e(w(1), w(1, 0)))) \\
& \quad = \pi(\Lambda(w(*, 0, 1)) \mid \Lambda(w(1)), \Lambda(e(w(1), w(1, 0)))) \\
& \quad = \pi(b_1 \mid a_1)
\end{aligned}$$

$$\begin{aligned}
& \pi(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1)) \mid \Lambda(w(1))) \\
& \quad = \pi(x_0 \mid a_1) \pi(b_1 \mid a_1) \\
& \quad \quad + \pi(x_1 \mid a_1) \pi(b_1 \mid a_1) \pi(c_0 \mid x_1 b_1) \\
& \quad \quad + \pi(x_1 \mid a_1) \pi(b_1 \mid a_1) \pi(c_1 \mid x_1 b_1) \\
& \quad = \pi(b_1 \mid a_1) \checkmark
\end{aligned}$$

So $\{\Lambda_z\}$ is a Back Door partition of the atomic events.

Again, it is unnecessary, but we can check that:

$$(A) \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$$

$$(B) \pi_{do\Lambda_x}(\Lambda_z) = \pi(\Lambda_z) \quad \text{for all } \Lambda_z$$

(A) Looking at $D_{do\Lambda_x}$ we get:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(0, *, 0, 0))) &= \pi(y_0 \mid a_0 b_0 c_0) \\ \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(0, *, 0, 1))) &= \pi(y_0 \mid a_0 b_0 c_1) \\ \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(1, *, 0))) &= \pi(y_0 \mid a_1 b_0) \\ \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1))) \\ &= \pi_{do\Lambda_x}(\Lambda_y \mid \Lambda(w(*, 0, 1))) \\ &= \pi(c_0 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_0) \\ &\quad + \pi(c_1 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_1) \\ &= \pi(y_0 \mid x_0 b_1) \end{aligned}$$

Looking at C we get:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda_x, \Lambda(w(0, *, 0, 0))) &= \pi(y_0 \mid x_0, \Lambda(w(0, *, 0, 0))) \\ &= \pi(y_0 \mid \Lambda(w(0, *, 0, 0))) \\ &= \pi(y_0 \mid a_0 b_0 c_0) \checkmark \end{aligned}$$

Similarly

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda_x, \Lambda(w(0, *, 0, 1))) &= \pi(y_0 \mid a_0 b_0 c_1) \checkmark \\ \pi(\Lambda_y \mid \Lambda_x, \Lambda(w(1, *, 0))) &= \pi(y_0 \mid x_0, \Lambda(w(1, *, 0))) \\ &= \pi(y_0 \mid \Lambda(w(1, *, 0))) \\ &= \pi(y_0 \mid a_1 b_0) \checkmark \end{aligned}$$

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda_x, \Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1))) \\ &= \pi(y_0 \mid x_0, \Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1))) \\ &= \pi(y_0 \mid \Lambda(w(*, 0, 1))) \\ &= \pi(c_0 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_0) \\ &\quad + \pi(c_1 \mid x_0 b_1) \pi(y_0 \mid x_0 b_1 c_1) \\ &= \pi(y_0 \mid x_0 b_1) \checkmark \end{aligned}$$

(B) Looking at $D_{do\Lambda_x}$ we get:

$$\begin{aligned}\pi_{do\Lambda_x}(\Lambda(w(0, *, 0, 0))) &= \pi(a_0) \pi(b_0 | a_0) \pi(c_0 | a_0 b_0) \\ &= \pi(a_0 b_0 c_0)\end{aligned}$$

$$\pi_{do\Lambda_x}(\Lambda(w(0, *, 0, 1))) = \pi(a_0 b_0 c_1)$$

$$\begin{aligned}\pi_{do\Lambda_x}(\Lambda(w(1, *, 0))) &= \pi(a_1) \pi(b_0 | a_1) \\ &= \pi(a_1 b_0)\end{aligned}$$

$$\begin{aligned}\pi_{do\Lambda_x}(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup (w(*, 1, 1, 1))) \\ &= \pi_{do\Lambda_x}(\Lambda(w(*, 0, 1))) \\ &= \pi(a_0) \pi(b_1 | a_0) + \pi(a_1) \pi(b_1 | a_1) \\ &= \pi(b_1)\end{aligned}$$

Looking at C we get:

$$\begin{aligned}\pi(\Lambda(w(0, *, 0, 0))) &= \pi(a_0) [\pi(x_0 | a_0) + \pi(x_1 | a_0)] \pi(b_0 | a_0) \pi(c_0 | a_0 b_0) \\ &= \pi(a_0 b_0 c_0) \checkmark\end{aligned}$$

Similarly

$$\begin{aligned}\pi(\Lambda(w(0, *, 0, 1))) &= \pi(a_0 b_0 c_1) \checkmark \\ \pi(\Lambda(w(1, *, 0))) &= \pi(a_1) [\pi(x_0 | a_1) + \pi(x_1 | a_1)] \pi(b_0 | a_1) \\ &= \pi(a_1 b_0) \checkmark\end{aligned}$$

$$\begin{aligned}\pi(\Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup (w(*, 1, 1, 1))) \\ &= \pi(a_0) \pi(x_0 | a_0) \pi(b_1 | a_0) \\ &\quad + \pi(a_1) \pi(x_0 | a_1) \pi(b_1 | a_1) \\ &\quad + \pi(a_0) \pi(x_1 | a_0) \pi(b_1 | a_0) \pi(c_0 | x_1 b_1) \\ &\quad + \pi(a_1) \pi(x_1 | a_1) \pi(b_1 | a_1) \pi(c_0 | x_1 b_1) \\ &\quad + \pi(a_0) \pi(x_1 | a_0) \pi(b_1 | a_0) \pi(c_1 | x_1 b_1) \\ &\quad + \pi(a_1) \pi(x_1 | a_1) \pi(b_1 | a_1) \pi(c_1 | x_1 b_1) \\ &= \pi(b_1) \checkmark\end{aligned}$$

We can therefore write:

$$\pi(y_0 | do x_0) = \pi(\Lambda_y | do \Lambda_x)$$

$$\begin{aligned}
&= \pi_{do\Lambda_x}(\Lambda_y) \\
&= \sum_z \pi(\Lambda_y \mid \Lambda_x, \Lambda_z) \pi(\Lambda_z) \\
&= \pi(y_0 \mid x_0, \Lambda(w(0, *, 0, 0))) \pi(\Lambda(w(0, *, 0, 0))) \\
&\quad + \pi(y_0 \mid x_0, \Lambda(w(0, *, 0, 1))) \pi(\Lambda(w(0, *, 0, 1))) \\
&\quad + \dots \\
&= \pi(y_0 \mid a_0 b_0 c_0) \pi(a_0 b_0 c_0) \\
&\quad + \pi(y_0 \mid a_0 b_0 c_1) \pi(a_0 b_0 c_1) \\
&\quad + \pi(y_0 \mid a_1 b_0) \pi(a_1 b_0) \\
&\quad + \pi(y_0 \mid x_0 b_1) \pi(b_1) \\
&= \pi(b_0) \pi(y_0 \mid b_0) + \pi(b_1) \pi(y_0 \mid x_0 b_1)
\end{aligned}$$

which is the result we got by looking at the $w_0 \rightarrow w_\infty$ paths in $D_{do\Lambda_x}$.

Now, we have just spent over two pages deriving our expression using our new Back Door Theorem, and it might appear that the process is overly cumbersome. In fact the majority of the working has been simply showing that this example fits the theory. All we actually need to do is:

- Produce $\{\Lambda_z\}$ as prescribed in section 5.5.5, and check that it satisfies our Back Door condition (5.5.1) (a little over half a page in this example)
- Substitute probabilities taken from our CEG C into our Back Door expression and simplify (nine lines in this example)

Notice that the full expression for $\pi(y_0 \mid do\ x_0)$ given above includes settings of the criterion-variable C , which would be impossible if we tried to use a BN for this model, as C is a descendant of X (see for example the edges leaving $w(*, 0, 1)$), and one of Pearl's conditions is that Z must contain no descendants of X . Notice also that the final expression only involves settings of the criterion-variable B , which would also be impossible if we tried to use a BN for this model, as in such a representation B would not block all Back Door paths from X to Y (Y is a child of A which is the parent of X).

Note that in this example we have three Λ_z which are singletons – they are the events associated with a single position. The fourth Λ_z is the union of the events associated with three positions, all of which contribute to the term $\pi(\Lambda_z)$ in our probability expression, but only one of which exists in $D_{do\Lambda_x}$ and hence contributes to the term $\pi_{do\Lambda_x}(\Lambda_y \mid \Lambda_z) = \pi(\Lambda_y \mid \Lambda_x, \Lambda_z)$.

Note also that when $\Lambda_z = \Lambda(w_z)$ for a single w_z ; if like in this example it is labelled as a sequence of criterion-variable-values (eg. $\Lambda(w(0, *, 0, 0)) = a_0 b_0 c_0$ etc.), then this sequence cannot contain a value of X if our Back Door condition (5.5.1) is to be satisfied for that Λ_z .

Now for the fourth Λ_z in this example, we see that the three positions whose associated events in union make up Λ_z are $w(*, 0, 1)$, $w(*, 1, 1, 0)$ and $w(*, 1, 1, 1)$, each of which contains a value for X . However $\Lambda_z = \Lambda(w(*, 0, 1)) \cup \Lambda(w(*, 1, 1, 0)) \cup \Lambda(w(*, 1, 1, 1))$ can simply be labelled as b_1 (A, B, C, X, Y all being binary), which does not contain a value for X , and hence it is possible that our Back Door condition (5.5.1) is satisfied for this Λ_z .

This gives us an insight into how to choose the component Λ_z of our partition:

If we can find w_z such that $\Lambda(w_z)$ satisfies:

$$\pi(\Lambda(w_z) \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) = \pi(\Lambda(w_z) \mid \Lambda(w_X)) \quad \forall w_X \in \{w_X\}$$

then we can make $\Lambda(w_z)$ a Λ_z .

Other Λ_z are produced by combining one position w_z that exists in $D_{do\Lambda_x}$ with other positions $\{w_z\}$ that disappear when we create $D_{do\Lambda_x}$, in such a way that the union of their associated events satisfies the Back Door condition (5.5.1) for all $w_X \in \{w_X\}$.

5.5.7 Blocking sets upstream of the manipulation

In choosing a Blocking set for Pearl's BN-based Back Door Theorem, we can always choose $Z = Q(X)$, the parents of X (provided we can calculate / estimate the relevant terms in our Back Door expression). Under certain circumstances we can choose Z to be more distant ancestors of X , and we can choose Z to be descendants of ancestors of X , provided that Z contains no descendants of X . In Example 2 for instance, $A = Q(X)$, and B is a child of A . In drawing our CEG for the model in Example 2, edges corresponding to A must precede those corresponding to X , but edges corresponding to B can precede or succeed those corresponding to X .

It will have been noted that the partition $\{\Lambda_z\}$ used in section 5.5.5 consisted of Λ_z of the form $\Lambda_z = \bigcup \Lambda(w_z)$ where each w_z lay downstream of the set $\{w_X\}$. Given the above, this might seem to be somewhat restrictive. So consider Λ_x, Λ_y as defined in sections 5.5.3 and 5.5.4, and $\{\Lambda_z\}$ a partition of the atomic events such that each Λ_z is of the form:

$$\Lambda_z = \bigcup_{i \in I} \Lambda(w_z^i)$$

where $\Lambda(w_z^i) \cap \Lambda(w_z^j) = \phi$ for $i, j \in I$, $i \neq j$, and where the set $\{w_z\}$ lies upstream of the set $\{w_X\}$.

Consider condition (5.5.1):

$$\begin{aligned} \pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) &= \frac{\pi(\Lambda_z, \Lambda(w_X), \Lambda(e(w_X, w'_X)))}{\pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)))} \\ &= \frac{\sum_{i \in I} \pi(\Lambda(w_z^i), \Lambda(w_X), \Lambda(e(w_X, w'_X)))}{\pi(\Lambda(w_X), \Lambda(e(w_X, w'_X)))} \end{aligned}$$

since $\Lambda(w_z^i) \cap \Lambda(w_z^j) = \phi$ for $i, j \in I$, $i \neq j$

$$= \frac{\sum_{i \in I} \pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_z^i), \Lambda(w_X)) \pi(\Lambda(w_z^i), \Lambda(w_X))}{\pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_X)) \pi(\Lambda(w_X))}$$

Since the summand equals zero if $w_z^i \not\prec w_X$ we need only concern ourselves with paths where $w_z^i \prec w_X \prec e(w_X, w'_X)$, so we have:

$$\pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) = \frac{\sum_{i \in I} \pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_X)) \pi(\Lambda(w_z^i), \Lambda(w_X))}{\pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_X)) \pi(\Lambda(w_X))}$$

using one of the class of results spawned by Proposition 5

$$\begin{aligned} &= \frac{\pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_X)) \sum_{i \in I} \pi(\Lambda(w_z^i), \Lambda(w_X))}{\pi(\Lambda(e(w_X, w'_X)) \mid \Lambda(w_X)) \pi(\Lambda(w_X))} \\ &= \frac{\sum_{i \in I} \pi(\Lambda(w_z^i), \Lambda(w_X))}{\pi(\Lambda(w_X))} \\ &= \frac{\pi(\Lambda_z, \Lambda(w_X))}{\pi(\Lambda(w_X))} \\ &= \pi(\Lambda_z \mid \Lambda(w_X)) \end{aligned}$$

so condition (5.5.1) is automatically satisfied for $\{\Lambda_z\}$ of this form.

Consider condition (5.5.2):

$$\begin{aligned} &\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) \\ &= \frac{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_y)}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \\ &= \frac{\sum_{i \in I} \pi(\Lambda(w_z^i), \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_y)}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \end{aligned}$$

Note that the summand equals zero if $w_z^i \not\prec w_{X(1)}$, so this equals:

$$\begin{aligned} &\frac{\sum_{i \in I} \pi(\Lambda_y \mid \Lambda(w_z^i), \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) \pi(\Lambda(w_z^i), \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \\ &= \frac{\sum_{i \in I} \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) \pi(\Lambda(w_z^i), \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \end{aligned}$$

using the specified form of Λ_y , the fact that $\{w_X\}$ are downstream of $\{w_z\}$, and one of

the class of results spawned by Proposition 5

$$\begin{aligned}
&= \frac{\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) \sum_{i \in I} \pi(\Lambda(w_z^i), \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \\
&= \frac{\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) \pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))}{\pi(\Lambda_z, \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))} \\
&= \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})))
\end{aligned}$$

Now suppose condition (5.5.2) is satisfied. Then:

$$\begin{aligned}
\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)})), \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)})), \Lambda_z) \\
\Rightarrow \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) &= \pi(\Lambda_y \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)}))) \\
\Rightarrow \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) &= \pi(\Lambda_y \mid \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))]) \\
&= \pi(\Lambda_y \mid \Lambda_x)
\end{aligned}$$

[since $\pi(A \mid B_1) = \pi(A \mid B_2)$ with $B_1 \cap B_2 = \phi \Rightarrow \pi(A \mid B_1) = \pi(A \mid B_1 \cup B_2)$]

So condition (5.5.2) implies that:

$$\begin{aligned}
\pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) &= \pi(\Lambda_y \mid \Lambda_x) \\
\Rightarrow \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x) &= \pi(\Lambda_y \mid \Lambda_x)
\end{aligned}$$

Now consider:

$$\pi_{do\Lambda_x}(\Lambda_y) = \sum_{w_X} \pi(\Lambda(w_X)) \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)))$$

from the proof in section 5.5.4.

If condition (5.5.2) holds, then this equals:

$$\begin{aligned}
&\sum_{w_X} \pi(\Lambda(w_X)) \pi(\Lambda_y \mid \Lambda_x) \\
&= \pi(\Lambda_y \mid \Lambda_x)
\end{aligned}$$

since $\{w_X\}$ forms a partition of the atomic events.

So, if condition (5.5.2) holds, then:

$$\pi_{do\Lambda_x}(\Lambda_y) = \pi(\Lambda_y \mid \Lambda_x)$$

Now, we have said (Proposition 13) that $\{\Lambda_z\}$ is a Back Door blocking set (partition) if conditions (5.5.1) and (5.5.2) hold for all $\Lambda_z \in \{\Lambda_z\}$, $w_X \in \{w_X\}$. If we choose $\{\Lambda_z\}$ of the form specified here then condition (5.5.1) always holds, but if (5.5.2) also holds then the effects of manipulating the system to the event Λ_x can be calculated simply by looking at the system conditioned on the event Λ_x . But if we can do this, there is no

need to look for a Back Door blocking set at all as we can find out all we need to know by conditioning on the event rather than manipulating to it. So it is not worthwhile to attempt to construct Back Door blocking sets $\{\Lambda_z\}$ upstream of a manipulation.

5.5.8 Using $\{w_X\}$ to create a blocking set

We have seen that it is not worthwhile to consider a blocking set upstream of the set $\{w_X\}$, but we can look at using the set $\{w_X\}$ itself to create our blocking set. This has a direct analogy with analysis on BNs, where it is always possible to replace Pearl's set Z by the set $Q(X)$ (the variable-parents of X) to give a revised Back Door expression:

$$P(Y = y \mid do\ X = x) = \sum_{q(x)} P(y \mid x\ q(x))\ P(q(x))$$

This blocking set $Z = Q(X)$ is not derived from the conditions $Z \perp\!\!\!\perp X \mid Q(X)$ and $Y \perp\!\!\!\perp Q(X) \mid (X, Z)$, and similarly our Back Door partition $\{\Lambda_z\}$ here is not derived from conditions (5.5.1) and (5.5.2). Recalling the analogy between $Q(X) = q(x)$ for BNs and $\Lambda(w_X)$ for the CEGs suggests we look at a partition $\{\Lambda_z\}$ where each Λ_z is of the form:

$$\Lambda_z^I = \bigcup_{i \in I} \Lambda(w_{X(i)})$$

for some collection $\{w_{X(i)}\}_{i \in I} \subset \{w_X\}$, where each $w_X \in \{w_X\}$ is such that $\Lambda(w_X)$ is an element of Λ_z^I for some I .

Using the proof in section 5.5.4, we can write:

$$\begin{aligned} \pi_{do\Lambda_z}(\Lambda_y) &= \sum_{w_X} \pi(\Lambda(w_X))\ \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X))) \\ &= \sum_I \sum_{i \in I} \pi(\Lambda(w_{X(i)}))\ \pi(\Lambda_y \mid \Lambda(w_{X(i)}), \Lambda(e(w_{X(i)}, w'_{X(i)}))) \\ &= \sum_I \sum_{i \in I} \pi(\Lambda_y \mid \Lambda(w_{X(i)}), \Lambda_x)\ \pi(\Lambda(w_{X(i)})) \\ &= \sum_I \sum_{i \in I} \left[\frac{\pi(\Lambda_x, \Lambda_y \mid \Lambda(w_{X(i)}))}{\pi(\Lambda_x \mid \Lambda(w_{X(i)}))} \right] \pi(\Lambda(w_{X(i)})) \end{aligned}$$

Now suppose we make a further stipulation about the sets $\{w_{X(i)}\}_{i \in I}$, and state that each Λ_z is of the form:

$$\Lambda_z^I = \bigcup_{w_{X(i)} \in u_X^I} \Lambda(w_{X(i)}) = \Lambda(u_X^I)$$

for some u_X^I (using Definition 18 from section 3.2), where each u_X^I is a stage, and the set $\{u_X^I\}$ form a U -cut (section 3.5) across the CEG.

Then, using the form specified for Λ_x and Proposition 3 in section 3.3, we can write, for $w_{X(i)} \in u_X^I$:

$$\pi(\Lambda_x \mid \Lambda(w_{X(i)})) = \pi(\Lambda_x \mid \Lambda(u_X^I)) = \pi(\Lambda_x \mid \Lambda_z^I)$$

So:

$$\begin{aligned} \pi_{do\Lambda_x}(\Lambda_y) &= \sum_I \sum_{i \in I} \left[\frac{\pi(\Lambda_x, \Lambda_y \mid \Lambda(w_{X(i)}))}{\pi(\Lambda_x \mid \Lambda_z^I)} \right] \pi(\Lambda(w_{X(i)})) \\ &= \sum_I \left[\frac{\sum_{i \in I} \pi(\Lambda_x, \Lambda_y \mid \Lambda(w_{X(i)})) \pi(\Lambda(w_{X(i)}))}{\pi(\Lambda_x \mid \Lambda_z^I)} \right] \\ &= \sum_I \left[\frac{\sum_{i \in I} \pi(\Lambda(w_{X(i)}), \Lambda_x, \Lambda_y)}{\pi(\Lambda_x \mid \Lambda_z^I)} \right] \\ &= \sum_I \left[\frac{\pi(\Lambda_z^I, \Lambda_x, \Lambda_y)}{\pi(\Lambda_x \mid \Lambda_z^I)} \right] \\ &= \sum_I \pi(\Lambda_y \mid \Lambda_z^I, \Lambda_x) \pi(\Lambda_z^I) \\ &= \sum_{u_X} \pi(\Lambda_y \mid \Lambda(u_X), \Lambda_x) \pi(\Lambda(u_X)) \end{aligned}$$

So, we can use the set $\{w_X\}$ to create a blocking set if we insist that each Λ_z is $\Lambda(u_X)$ for some stage (or set of stage-equivalent positions) u_X . This of course requires our set $\{w_X\}$ to be organised into stages $\{u_X\}$, which is not an onerous restriction.

Note that, although it may be possible to create a partition where $\{\Lambda_z\}$ are not of the form $\Lambda_z = \Lambda(u_X)$, the proof above relies on this form so that we can replace $\pi(\Lambda_x \mid \Lambda(w_{X(i)}))$ by $\pi(\Lambda_x \mid \Lambda(u_X^I)) = \pi(\Lambda_x \mid \Lambda_z^I)$.

Example 4:

Consider the BN and CEG from Example 2, and the manipulation $do \Lambda_x = do (X = x_0)$.

Let $\{\Lambda_z\} = \{\Lambda(u_X)\}$ with:

$$u_X^1 = \{w(0,0), w(0,1)\}$$

$$u_X^2 = \{w(1,0), w(1,1)\}$$

Then:

$$\begin{aligned} \pi(\Lambda_z^1) &= \pi(\Lambda(u_X^1)) \\ &= \pi(\Lambda(w(0,0)) \cup \Lambda(w(0,1))) \\ &= \pi(a_0) \\ \pi(\Lambda_z^2) &= \pi(a_1) \end{aligned}$$

Also:

$$\begin{aligned}\pi(\Lambda_y \mid \Lambda_z^1, \Lambda_x) &= \pi(y_0 \mid \Lambda(w(0,0)) \cup \Lambda(w(0,1)), x_0) \\ &= \pi(y_0 \mid a_0, x_0) \\ \pi(\Lambda_y \mid \Lambda_z^2, \Lambda_x) &= \pi(y_0 \mid a_1, x_0)\end{aligned}$$

giving:

$$\begin{aligned}\pi_{do\Lambda_x}(\Lambda_y) &= \pi(y_0 \mid do\ x_0) \\ &= \sum_a \pi(y_0 \mid a, x_0) \pi(a)\end{aligned}$$

exactly as we'd get if we used Pearl's expression on the BN in Figure 2, and $Z = Q(X) = \{A\}$.

5.5.9 Some thoughts on general Singular manipulations

We have so far restricted ourselves to Singular manipulations of the form $do\ (X = x)$ for some criterion variable X . If we were instead to let $\{w_X\}$ be any W -cut of our CEG C , and the edges $\{e(w_X, w'_X)\}$ simply be a set of specified edges leaving $\{w_X\}$, would our theory still hold?

Well, Λ_x would still have the form specified in section 5.5.3 (although $e(w_X, w'_X)$ would no longer necessarily be an edge labelled x_0). The equivalence between $w'_X \rightarrow w_\infty$ subpaths in D_{Λ_x} and $D_{do\Lambda_x}$ would not be affected, and our conditions (5.5.1) and (5.5.2) would remain unchanged. Proposition 13 would still be valid.

Similarly, none of the working in section 5.5.5 requires the positions $\{w_X\}$ all to have criterion variable X , nor $\{e(w_X, w'_X)\}$ to be the set of edges labelled x_0 .

If we consider the ideas of section 5.5.8, clearly for any stage u_X in our Back Door partition, we must have each $w_X \in u_X$ having the same criterion variable, and each edge $e(w_X, w'_X)$ carrying the same label, but there is nothing to stop us having different u_X in our partition having different criterion variables.

So, we can extend the theory to general Singular manipulations, which means that the CEG here is far more flexible than the BN. We have shown in Example 3 that we can analyse manipulations of asymmetric models, and also that we can use what might be termed asymmetric blocking sets in a CEG-based Back Door Theorem. But extending the theory to general Singular manipulations means that we can also analyse the effects of

asymmetric manipulations, which cannot realistically be analysed with a BN – in a BN one cannot analyse interventions where both the manipulated-variable and the manipulated-variable-value can differ for different settings of the other variables.

Note also that it would not be difficult for us to create a Back Door partition that consisted of some positions $\{w_z\}$ downstream of the manipulation together with some stages $\{u_X\}$ coincident with the manipulation. Recall that a good Back Door expression allows the analyst to estimate the probabilities of effects of a cause from a partially observed system, so this flexibility in our choice of partition set is very useful when some of the events in the system are unobservable or have large observational costs. Standard causal analysis with BNs requires one to be able to calculate or estimate $P(z)$ and $P(y_0 | x_0 z)$ for all values z of the blocking set of variables Z . This is not necessary with CEGs – Our blocking sets do not need to correspond to any fixed subset of the measurement random variables that define a BN. We have also seen (in Example 3) that we can use the CEG-based version of the Back Door Theorem in cases where it would be impossible to use the BN-based version, as the model does not obey the conditions specified by Pearl.

5.6 A Front Door Theorem for CEGs

5.6.1 Background

As well as the Back Door Theorem, Pearl [41] has also produced a Front Door Theorem, which can be used in cases where the Back Door Theorem conditions do not hold or where the events needing to be observed for the Back Door Theorem have too large an observational cost. Like the Back Door Theorem, the Front Door Theorem allows one to reduce the complexity of the general manipulated-probability expression used with BNs, and can allow one to sidestep identifiability problems associated with it.

Pearl's Front Door Theorem states that under certain conditions on sets of variables X, Y, Z , we can write:

$$P(Y = y_0 | do X = x_0) = \sum_z P(z | x_0) \sum_x P(y_0 | x z) P(x)$$

an expression whose value can be estimated from a partially observed idle system.

Pearl quotes three conditions for using the Front Door Theorem, and these can be reduced to two conditional independence conditions:

$$Y \perp\!\!\!\perp X | (Z, Q(X))$$

$$Z \perp\!\!\!\perp Q(X) | X$$

NB: I have not seen these two conditions expressed anywhere else, although it seems unlikely that researchers such as Dawid and Lauritzen are unaware of them.

Using the same approach as in section 5.5.4, we can suggest appropriate CEG-versions of these conditions. We confine ourselves here to Singular manipulations, and define Λ_x as in section 5.5.3 and Λ_y as in section 5.5.4. Recall also that $\Lambda(w_X)$ takes the role of $Q(X) = q(x)$ in our conditional independence conditions. We let $\{\Lambda_z\}$ be a partition of the atomic $w_0 \rightarrow w_\infty$ paths, and impose no further restrictions on the form of Λ_z (as for example is done in section 5.5.5). Using the analogies from section 5.5.4 we would replace our two conditional independence conditions by:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda(w_X), \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))], \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x, \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \\ \Rightarrow \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w'_X)), \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \end{aligned} \quad (5.6.1)$$

and

$$\begin{aligned} \pi(\Lambda_z \mid \Lambda(w_X), \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))]) &= \pi(\Lambda_z \mid \Lambda(w_X), \Lambda_x) \\ &= \pi(\Lambda_z \mid \bigcup_{w_X} [\Lambda(w_X), \Lambda(e(w_X, w'_X))]) \\ &= \pi(\Lambda_z \mid \Lambda_x) \\ \Rightarrow \pi(\Lambda_z \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w'_{X(1)}))) &= \pi(\Lambda_z \mid \Lambda_x) \\ &= \pi(\Lambda_z \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w'_{X(2)}))) \end{aligned} \quad (5.6.2)$$

where $w_{X(1)}$, $w_{X(2)}$ are different positions in our set $\{w_X\}$.

But our Front Door expression, if it is anything like Pearl's, will require us to sum over some variable corresponding to Pearl's X . So we need to produce a partition of the atomic events, of which Λ_x is one element. This in turn is going to require us to express conditions (5.6.1) and (5.6.2) in such a way that they refer to all elements of this partition, rather than just Λ_x .

In order to do this, we restrict ourselves to the type of Singular manipulation described in section 5.5.3 – interventions of the form $do(X = x_i)$ for some criterion variable X , as opposed to the general Singular manipulations described in section 5.5.9. It is therefore not unreasonable to make the assumption that every position $w_X \in \{w_X\}$ has the

same number of outgoing edges, and that these edges represent the same *outcomes* for each w_X (ie. each w_X has an outgoing edge labelled x_i for each $x_i \in \{x_i\}_{i \in I}$). Hence we can partition the atomic events as:

$$\{\Lambda_x^i\}_{i \in I} = \left\{ \bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w_X^i))] \right\}_{i \in I}$$

where the edge $e(w_X, w_X^i)$ is the edge leaving w_X labelled x_i .

We can now re-express our conditions (5.6.1) and (5.6.2) as:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x^i, \Lambda_z) &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w_X^i)), \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w_X^j)), \Lambda_z) \\ &= \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x^j, \Lambda_z) \end{aligned} \tag{5.6.3}$$

for all $w_X \in \{w_X\}$.

$$\begin{aligned} \pi(\Lambda_z \mid \Lambda(w_{X(1)}), \Lambda_x^i) &= \pi(\Lambda_z \mid \Lambda(w_{X(1)}), \Lambda(e(w_{X(1)}, w_{X(1)}^i))) \\ &= \pi(\Lambda_z \mid \Lambda_x^i) \\ &= \pi(\Lambda_z \mid \Lambda(w_{X(2)}), \Lambda(e(w_{X(2)}, w_{X(2)}^i))) \\ &= \pi(\Lambda_z \mid \Lambda(w_{X(2)}), \Lambda_x^i) \end{aligned} \tag{5.6.4}$$

for all $\Lambda_x^i \in \{\Lambda_x^i\}$.

5.6.2 A Front Door Theorem for Singular manipulations

Proposition 14:

If Λ_x, Λ_y are defined as in sections 5.5.3 and 5.5.4, and $\{\Lambda_z\}$ is a partition of the $w_0 \rightarrow w_\infty$ paths in C that satisfies conditions (5.6.3) and (5.6.4) above, then $\{\Lambda_z\}$ is a Front Door partition of the $w_0 \rightarrow w_\infty$ paths, and:

$$\pi_{do\Lambda_z}(\Lambda_y) = \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_i \pi(\Lambda_y \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i)$$

Proof:

This follows the proof of Proposition 13 until the line:

$$\begin{aligned} &\pi_{do\Lambda_z}(\Lambda_y) \\ &= \sum_{w_X} \pi(\Lambda(w_X)) \sum_z \pi(\Lambda_y \mid \Lambda(w_X), \Lambda(e(w_X, w_X')), \Lambda_z) \pi(\Lambda_z \mid \Lambda(w_X), \Lambda(e(w_X, w_X'))) \end{aligned}$$

We then invoke conditions (5.6.3) and (5.6.4) to give:

$$\begin{aligned}
\pi_{do\Lambda_x}(\Lambda_y) &= \sum_{w_X} \pi(\Lambda(w_X)) \sum_z \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \pi(\Lambda_z \mid \Lambda_x) \\
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \pi(\Lambda(w_X)) \\
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \sum_i \pi(\Lambda(w_X), \Lambda_x^i)
\end{aligned}$$

since $\{\Lambda_x^i\}$ forms a partition of the atomic events

$$\begin{aligned}
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \sum_i \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_z) \pi(\Lambda(w_X), \Lambda_x^i) \\
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \sum_i \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x^i, \Lambda_z) \pi(\Lambda(w_X), \Lambda_x^i)
\end{aligned}$$

using condition (5.6.3)

But:

$$\begin{aligned}
\pi(\Lambda(w_X), \Lambda_x^i) &= \frac{\pi(\Lambda(w_X), \Lambda_x^i, \Lambda_z)}{\pi(\Lambda_z \mid \Lambda(w_X), \Lambda_x^i)} \\
&= \frac{\pi(\Lambda(w_X), \Lambda_x^i, \Lambda_z)}{\pi(\Lambda_z \mid \Lambda_x^i)}
\end{aligned}$$

using condition (5.6.4)

$$\begin{aligned}
&= \frac{\pi(\Lambda(w_X), \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i)}{\pi(\Lambda_x^i, \Lambda_z)} \\
&= \pi(\Lambda(w_X) \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i)
\end{aligned}$$

So:

$$\begin{aligned}
\pi_{do\Lambda_x}(\Lambda_y) &= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \sum_i \pi(\Lambda_y \mid \Lambda(w_X), \Lambda_x^i, \Lambda_z) \pi(\Lambda(w_X) \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i) \\
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_{w_X} \sum_i \pi(\Lambda(w_X), \Lambda_y \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i) \\
&= \sum_z \pi(\Lambda_z \mid \Lambda_x) \sum_i \pi(\Lambda_y \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i)
\end{aligned}$$

□

Note that we may well choose to define $\Lambda_z = \bigcup \Lambda(w_z)$ where $\{w_z\}$ form a W -cut across the CEG downstream of $\{w_X\}$ and upstream of the edges labelled y_0 . If we do so, then the fact that $\{w_X\}$ precedes $\{e(w_X, w_X^i)\}$ precedes $\{w_z\}$, does not prevent us from writing expressions like $\pi(\Lambda(w_X) \mid \Lambda_x^i, \Lambda_z)$, since all our events are precisely defined as the unions of $w_0 \rightarrow w_\infty$ paths in C .

Example 5:

Consider the BN and corresponding CEG in Figure 23. Consider also the manipulation $do \Lambda_x = do (X = x_0)$, the event $\Lambda_y = (Y = y_0)$, and the expression $\pi(\Lambda_y \mid do \Lambda_x) = \pi(y_0 \mid do x_0)$. Our graph $D_{do\Lambda_x}$ is given in Figure 24.

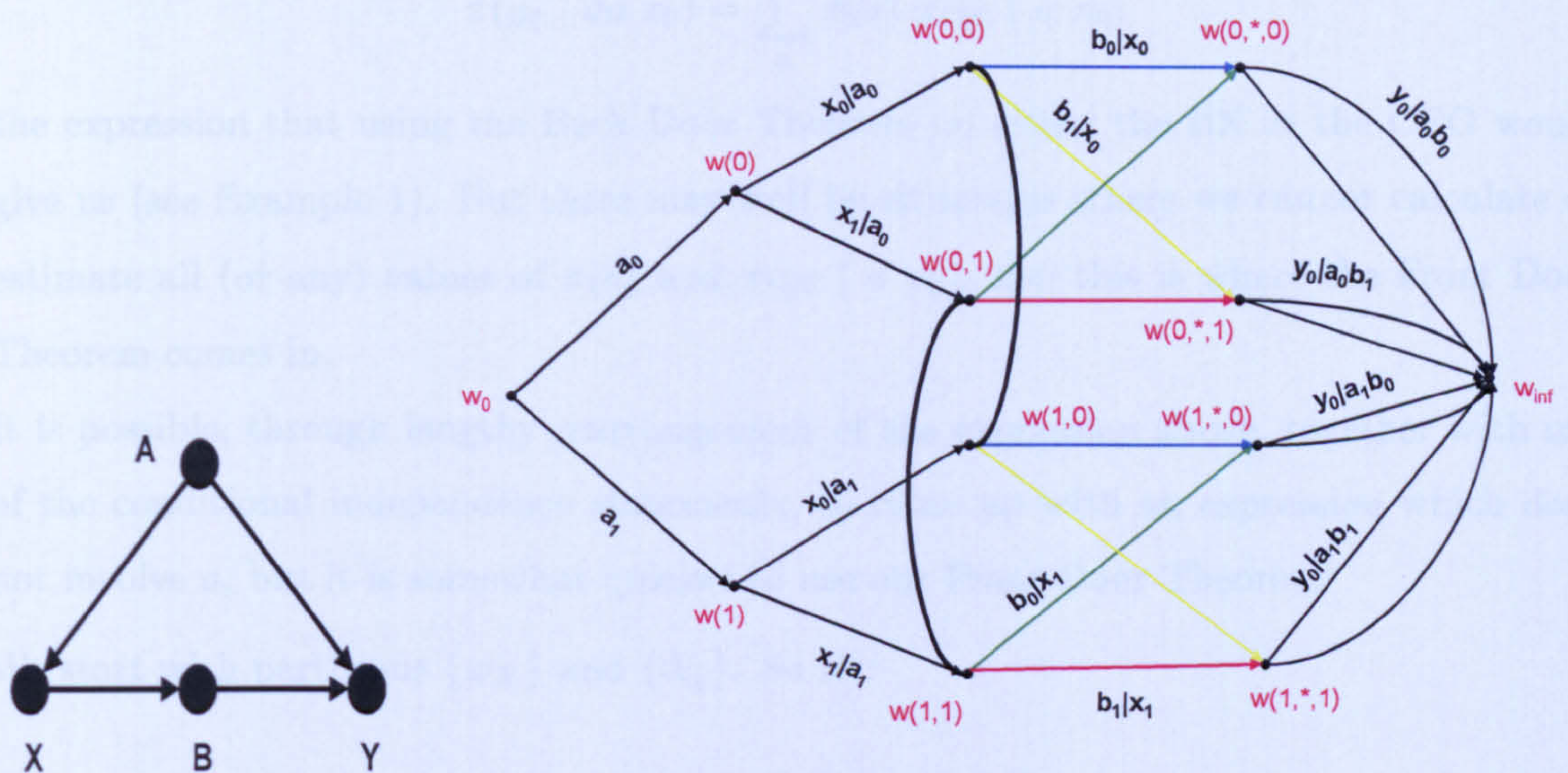


Figure 23: Bayesian Network and CEG for Example 5

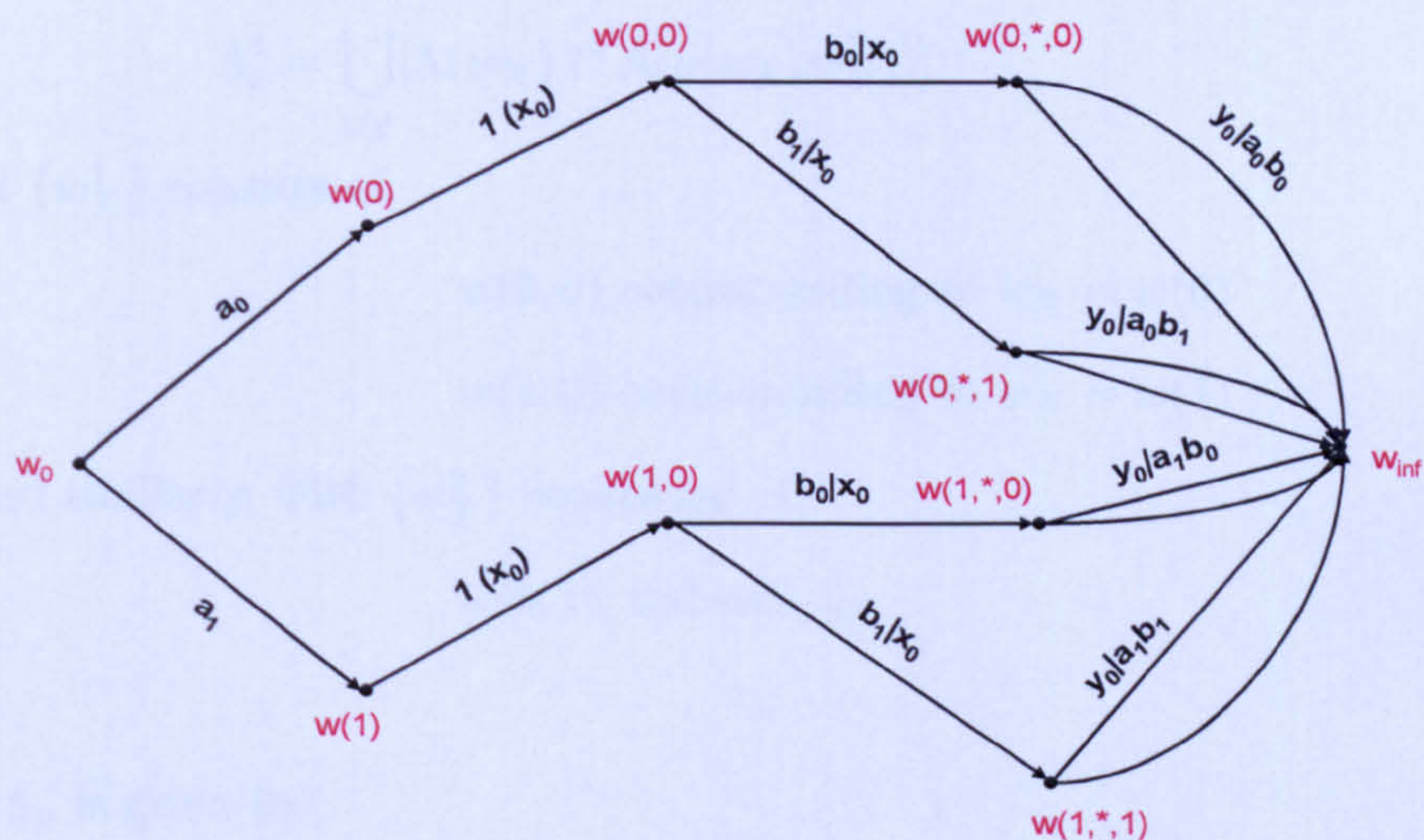


Figure 24: Derived Graph $D_{do\Lambda_x}$ for Example 5

The model and manipulation represented here can be analysed using both the Back Door Theorem (see Example 1) and the Front Door Theorem. The conditional independence properties of the model, readable from either the BN or the CEG are $B \perp\!\!\!\perp A \mid X$ and $Y \perp\!\!\!\perp X \mid (A, B)$. The expression for $\pi(y_0 \mid do\ x_0)$ readable straight off $D_{do\Lambda_x}$ or the BN is:

$$\pi(y_0 \mid do\ x_0) = \sum_a \pi(a) \sum_b \pi(b \mid x_0) \pi(y_0 \mid a\ b)$$

A little rearrangement of this expression using the two conditional independence state-

ments gives us:

$$\pi(y_0 \mid do\ x_0) = \sum_a \pi(a) \pi(y_0 \mid a\ x_0)$$

the expression that using the Back Door Theorem on either the BN or the CEG would give us (see Example 1). But there may well be situations where we cannot calculate or estimate all (or any) values of $\pi(a)$ and $\pi(y_0 \mid a\ x_0)$, and this is where the Front Door Theorem comes in.

It is possible, through lengthy rearrangement of the expression above, together with use of the conditional independence statements, to come up with an expression which does not involve a , but it is somewhat quicker to use our Front Door Theorem!

We start with partitions $\{w_X\}$ and $\{\Lambda_x^i\}$. So let:

$$\{w_X\} = \{w(0), w(1)\}$$

$$\{\Lambda_x^i\} = \{\Lambda_x^1, \Lambda_x^2\}$$

where

$$\Lambda_x^1 = \bigcup_{w_X} [\Lambda(w_X) \cap \Lambda(e(w_X, w_X^1))]$$

and the set $\{w_X^1\}$ consists of

$$w(0, 0) \text{ corresponding to } w_X = w(0)$$

$$w(1, 0) \text{ corresponding to } w_X = w(1)$$

Λ_x^2 is defined similarly, with $\{w_X^2\}$ consisting of

$$w(0, 1) \text{ and } w(1, 1)$$

Our event Λ_y is given by:

$$\Lambda_y = \bigcup_{w_Y} [\Lambda(w_Y) \cap \Lambda(e^u(w_Y, w_\infty))]$$

where our $\{w_Y\}$ are:

$$\{w(0, *, 0), w(0, *, 1), w(1, *, 0), w(1, *, 1)\}$$

and $e^u(w_Y, w_\infty)$ is the (upper) edge from w_Y to w_∞ labelled y_0 .

In this example, we choose each Λ_z to have a form very similar to that of Λ_x^i or Λ_y (and different from the forms chosen in previous examples). So we let:

$$\Lambda_z^1 = \bigcup_{w_Z} [\Lambda(w_Z) \cap \Lambda(e(w_Z, w_Z^1))]$$

where our $\{w_Z\}$ are:

$$\{w(0,0), w(0,1), w(1,0), w(1,1)\}$$

and the set $\{w_Z^1\}$ consists of

$$w(0,*,0) \text{ corresponding to } w_Z = w(0,0), w(0,1)$$

$$w(1,*,0) \text{ corresponding to } w_Z = w(1,0), w(1,1)$$

Λ_z^2 is defined similarly, with $\{w_Z^2\}$ consisting of

$$w(0,*,1) \text{ corresponding to } w_Z = w(0,0), w(0,1)$$

$$w(1,*,1) \text{ corresponding to } w_Z = w(1,0), w(1,1)$$

We need to check that our partition satisfies conditions (5.6.3) and (5.6.4). We could do this through a moderately short algebraic process, but it is more satisfying to see if we can use the graph to check – this is quicker, and more in line with the checking of Pearl's conditions, which are expressed on the topology of the BN.

(5.6.3)

Consider the expression $\pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda_x^1, \Lambda_z^1)$. Now $\Lambda(w_{X(1)}) \cap \Lambda_z^1$ requires that we pass through $w(0)$ and $w(0,*,0)$, irrespective of which of Λ_x^1 or Λ_x^2 occurs.

Noting the form of Λ_y (and invoking one of the class of results spawned by Proposition 5), we have:

$$\begin{aligned} \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda_x^1, \Lambda_z^1) &= \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda_z^1) \\ &= \pi(\Lambda_y \mid \Lambda(w_{X(1)}), \Lambda_x^2, \Lambda_z^1) \end{aligned}$$

and by the symmetry of our CEG, (5.6.3) holds.

(5.6.4)

Consider the expression $\pi(\Lambda_z^1 \mid \Lambda(w_X), \Lambda_x^1)$. Now Λ_x^1 requires that we pass through $w(0,0)$ or $w(1,0)$; and $\Lambda_x^1 \cap \Lambda_z^1$ requires that we pass through either $w(0,0)$ and $w(0,*,0)$, or $w(1,0)$ and $w(1,*,0)$. But the existence of an undirected edge joining $w(0,0)$ to $w(1,0)$ means that:

$$\pi_e(w(0,*,0) \mid w(0,0)) = \pi_e(w(1,*,0) \mid w(1,0))$$

Noting the form of Λ_z (and invoking one of the class of results spawned by Proposition 5), we have:

$$\begin{aligned} \pi(\Lambda_z^1 \mid \Lambda(w_{X(1)}), \Lambda_x^1) &= \pi(\Lambda_z^1 \mid \Lambda_x^1) \\ &= \pi(\Lambda_z^1 \mid \Lambda(w_{X(2)}), \Lambda_x^1) \end{aligned}$$

and by the symmetry of our CEG, (5.6.4) holds.

The ease with which we can check these conditions on the graph suggests that, as with the Back Door Theorem, there are graphical versions of conditions (5.6.3) and (5.6.4), but finding these is a task for the future.

Conditions (5.6.3) and (5.6.4) having been satisfied, we can substitute into the expression from Proposition 14 to get the Front Door expression for this example. Again, we could do this by a (somewhat longer) algebraic process, but it is quicker and more satisfying to do it straight from the graph.

Our initial partitions were not chosen arbitrarily – the event Λ_x^1 is the union of the $w_0 \rightarrow w_\infty$ paths passing through edges labelled x_0 , and we can (as per chapter 2) give the event Λ_x^1 the label x_0 without ambiguity. Similarly we can label Λ_x^2 by x_1 , Λ_y by y_0 , Λ_z^1 by b_0 , and Λ_z^2 by b_1 . Substituting these into the expression:

$$\pi_{do\Lambda_x^1}(\Lambda_y) = \sum_z \pi(\Lambda_z \mid \Lambda_x^1) \sum_i \pi(\Lambda_y \mid \Lambda_x^i, \Lambda_z) \pi(\Lambda_x^i)$$

we get:

$$\pi(y_0 \mid do\ x_0) = \sum_b \pi(b \mid x_0) \sum_x \pi(y_0 \mid x\ b) \pi(x)$$

an expression which does not involve a .

5.6.3 Some further thoughts on the Front Door Theorem

Clearly, the development of the theory associated with a CEG-based version of the Front Door Theorem is not as far advanced as that associated with a CEG-based version of the Back Door Theorem. The most immediate of the tasks involved in the further development of the theory is the production of graphical versions of the conditions (5.6.3) and (5.6.4), similar to the graphical condition for the Back Door Theorem developed in section 5.5.5. The checking of condition (5.6.4) in Example 4 suggests that the graphical version of this condition may involve the undirected edges of the CEG C .

A second task is to look at a re-expression of the Front Door expression, analogous to the alternative version available for BNs:

$$\pi(y_0 \mid do\ x_0) = \sum_z \pi(z \mid do\ x_0) \pi(y_0 \mid do\ z)$$

The obvious equivalent expression for CEGs would be:

$$\pi_{do\Lambda_x}(\Lambda_y) = \sum_z \pi_{do\Lambda_x}(\Lambda_z) \pi_{do\Lambda_z}(\Lambda_y)$$

but this expression is not as straightforward as it looks, involving as it does, probabilities evaluated on several different graphs: $D_{do\Lambda_z}$ and $\{D_{do\Lambda_z}\}$.

5.7 Some final thoughts on the Causal manipulation of CEGs

In sections 5.5.4 through 5.5.9, whilst considering singular manipulations and the Back Door Theorem, we considered a partition $\{\Lambda_z\}$ which was *fixed* in the sense that its membership was constant. Causal analysis with CEGs, flexible as it already is, would become considerably more so if we could let the membership of $\{\Lambda_z\}$ depend in some way on whichever $w_X \in \{w_X\}$ we have passed through on our $w_0 \rightarrow w_\infty$ path. The problem here would be in interpreting and satisfying condition (5.5.2), and it might be more sensible to go back to the original conditions (A) and (B) for the Back Door Theorem, rather than try and adapt conditions (5.5.1) and (5.5.2) to fit this situation.

It would also be useful to go beyond Singular manipulations, and look at:

- Interventions where some $w_0 \rightarrow w_\infty$ paths have no edges manipulated – this would correspond, for example, to treatment regimes where only patients with certain combinations of symptoms (ie. at certain positions or stages) are treated.
- Interventions where some $w_0 \rightarrow w_\infty$ paths are subject to more than one manipulation.
- Interventions which impose a probability distribution on the children of our positions $\{w_X\}$, rather than just assigning a probability of 1 to one child. If, for some positions, this distribution was identical to that in the *idle* system, then we would have interventions of the sort described in the first bullet point.
- Interventions which produce possible outcomes at a position which are not possible in the idle system. This would involve not just imposing a new distribution on existing edges, but the adding of extra edges and hence the production of extra paths not present in the original CEG. If we enact the intervention *Build a dam across the valley mouth*, then the event *The village halfway up the valley side gets flooded*, which has zero probability in the idle system, now has a probability greater than zero.

Acceptance of the possibility of interventions as described in this last bullet point creates problems with our definition of a CEG – how can $C_{do\Lambda}$ contain paths that are not atoms or the unions of atoms from C ? One possible solution here would be to allow edges in a CEG to have zero probabilities. Then, in the last bullet point, we would simply have interventions that revealed previously hidden (ie. zero-probability) edges and paths.

Manipulations of the sort described in our second bullet point bring causal analysis on CEGs very close to Decision analysis on Decision Trees – such a Tree can be thought of

as an Event Tree where most paths are subject to a number of manipulations.

Throughout this chapter we have followed Pearl [41] in looking at the idea of manipulations, and analysing the effects of such manipulations. Clearly however, we can think of effects having causes without the need of a manipulation, and researchers in other fields sometimes attempt to analyse cause and effect without reference to an intervention. An interesting viewpoint is provided by Coggan and Martyn [8] – they tacitly accept the meaning of cause put forward by Pearl and others, and detailed in section 5.1, but point out that a cause of some effect will itself have causes, and that there might, in theory, exist an infinite causal chain.

We could interpret Pearl's *do* or our Singular manipulations as the breaking of this infinite chain at some point of our choosing. This links nicely with the idea expressed in section 5.1 that we can choose the level of detail we include in our representations and that this will depend on what we intend to *do* to the system.

I have not in this chapter looked at the process of abduction – Forsyth [13] says that we may abduce that a friend has a cold from her symptoms via the implication that colds (can) cause these symptoms, but also that we might discover the friend to have pneumonia by further questioning. By enacting an intervention we have in essence put a cause in place; and have then looked at its effects. We have not looked at deducing causes from effects – observing an event and considering the possible causes of this event.

6. Conclusion

6.1 Summary

The CEG is a graphical model specifically created to embody the conditional independence structure of problems whose state spaces are asymmetric and do not have a natural Product Space structure. They are both a representation of such problems, and a tool for their analysis. They have been developed to model the types of process that occur in risk analysis [2], biological regulation [7], emergency support systems [54], analysis of forensic evidence [43] etc., which are more readily represented as Trees than BNs.

BNs are the most commonly used graphical models for representing discrete problems, but become a rather clumsy tool when the problem being analysed has asymmetries of the sort that occur in our Blood condition and Machine monitoring examples (section 2.6). Attempts have been made to adapt the BN [3][16][34][59] to improve the analyst's ability to use it to represent and analyse asymmetric problems, but these have only been partially successful. The CEG, being derived from a tree, and having an event-based as opposed to variable-based topology, is a much more appropriate graphical model for such problems.

Chapter 2 of this thesis contains a detailed description of how to construct CEGs from Trees. In chapters 3 to 5 I have provided a collection of important results for CEGs in the areas of Reading, Propagation and Causal analysis. In each chapter I have illustrated how these results work, related the results to similar ones for BNs, attempted to explain their significance, and suggested further lines of related enquiry.

In particular, in chapter 3 I have created a semantic structure for the reading of CEGs, allowing me to derive three important results:

$$X(u) \perp\!\!\!\perp Z(u) \mid \Lambda(u)$$

$$Y(w) \perp\!\!\!\perp Z(w) \mid \Lambda(w)$$

and

$$\Lambda(w_3) \perp\!\!\!\perp \Lambda(w_1) \mid \Lambda(w_2) \quad \text{for } w_1 \prec w_2 \prec w_3$$

These are context-specific conditional independence statements which enable us to learn much more about the independence structure of a problem than we can with BNs, or even context-specific BNs. The first two statements tell us that if know at what

stage (u) or position (w) we are in a process, then the distribution of the possible immediately following steps ($\pi_{X(u)}$), or the distribution of the whole possible further development of the process ($\pi_{Y(w)}$), is determined solely by the history of the process up to that stage ($\Lambda(u)$) or position ($\Lambda(w)$). Our third result tells us that reaching a position (w_3) given that we have passed through an earlier position (w_2) is independent of the path taken to that earlier position.

These are powerful results, but there is still work to be done in this area, and in particular we will be looking to develop a d-separation theorem over the next few years, analogous to the d-separation theorem for BNs [41][30].

In chapter 4 I developed an algorithm for updating the topology of, and probability distribution over a CEG following an observation of a general event Λ . This algorithm can be expressed in a form that makes it directly analogous to the Junction Tree and Local Message Passing algorithms used with BNs. I also looked at possible *lazy shortcuts* in the algorithm, analogous to those used in Lazy propagation on BNs [35][6], which increase the speed of the propagation algorithm.

My intention now is to use this algorithm on real problems, and in particular to see how successful it is with high-dimensional asymmetric problems – we believe our algorithm will be more efficient than BN-based algorithms when used with asymmetric problems (see [63]). If this is so, it would be interesting to discover at what point in the transition from symmetric to asymmetric problems, our algorithm starts to become more efficient. As indicated in section 4.9, another area we will be looking at is dynamic propagation on CEGs, and trying to develop the time-sliced dynamic CEG as a means of representing and propagating on dynamic asymmetric processes. We also wish to develop a theory of *Learning CEGs* analogous to that for Learning BNs.

In chapter 5 I developed the theory of Causal manipulation of CEGs, and noted the similarity between the processes of conditioning on an event and manipulating to an event. I then introduced the idea of a singular manipulation – an intervention in which every $w_0 \rightarrow w_\infty$ path in C passes through one of a set of positions $\{w_X\}$, and where the event Λ_x simply assigns a probability of 1 to one edge leaving each of the positions in $\{w_X\}$.

I introduced a general Back Door Theorem for manipulations on a CEG, analogous to Pearl's [41], and then concentrated on a Back Door Theorem for singular manipulations, showing how the two conditions for this theorem related to Pearl's two conditions, and

how I could express one of the conditions graphically (on the topology of the CEG). I also produced a Front Door Theorem for singular manipulations analogous to Pearl's [41].

Section 5.7 suggested a number of areas for future development of the theory, but in particular it would be useful to be able to express all the conditions for the Back Door and Front Door theorems for singular manipulations graphically, and to be able to develop easy to check (ideally graphical) conditions for the Back Door and Front Door theorems for general manipulations. It is also worth noting that the process by which I derived a graphical version of condition (5.5.2) may help us in producing graphical conditions for a CEG-based d-separation theorem.

In sections 6.2 and 6.3 I look briefly at two areas I researched in the early years of my PhD, and suggest how they might be further investigated in the future. In section 6.4 I bring together ideas for future research not referred to in the earlier sections of the chapter.

6.2 (Causal) Estimation on CEGs

In [45], J.Q. Smith has started to look at the topic of estimation on CEGs – in particular conjugate estimation analogous to the conjugate prior to posterior analysis for BNs given ancestral data. This area of estimation is one we are likely to conduct further research into over the next few years. In this section however, I concentrate on two ideas, connected with estimation of causal probabilities, that I looked at briefly a couple of years ago, and suggest how they could be developed.

In [30], Lauritzen looks at the model represented by the BN in Figure 17 of section 5.5.1, and discusses the relative efficiency of the estimations of the general manipulated-probability expression (as given in section 5.5.1) given by the Back Door and Front Door expressions. He produces maximum likelihood estimators for these two expressions, as well as for the general manipulated-probability expression, and indicates that they are not equally efficient when estimated from data – there is a *loss of information associated with not observing all four variables*.

Lauritzen suggests that it *would be interesting to compare the loss of efficiency by not observing (a) versus not observing (b)*. Now a CEG, with its path-based topology, is a very easy structure on which to simulate data. I ran a primitive simulation to answer this question, which suggested that in the above model the Front Door estimate is more efficient than the Back Door – the mean estimate was closer to the value produced by the

full manipulated-probability expression, and the standard deviation of the estimates was lower than for the Back Door expression.

It would be interesting to investigate this theoretically and find out whether this result is generally true. It is worth noting that one needs to collect more data to produce a Front Door estimate, so it would be reasonable for this estimate to be a better one.

Now this is a task that could be tackled with a BN, but as already mentioned, the structure of the CEG lends itself to questions of this sort. The Back Door and Front Door expressions for a BN (and indeed nearly all such expressions used for estimating probabilities) require observations of values of the measurement variables of the BN. The CEG is a more flexible construct, and it is likely that our CEG-based expressions need only use specific functions of the data – indicators for intrinsic events may be sufficient. The CEG would also allow us to compare the relative efficiency of our CEG-based Back Door and Front Door expressions (from chapter 5) for asymmetric manipulations of asymmetric problems.

A similar question (looking at the efficiency of estimations of causal probabilities) was investigated by Kuroki and Miyakawa in [28]. They compared the relative efficiency for different choices of Back Door blocking sets when one uses a BN-representation of a model, multivariate Normal distributions and Pearl’s Back Door Theorem.

They state that if we have the choice between two blocking sets $\{W, Z_1\}$ and $\{W, Z_2\}$, where:

$$\begin{aligned} X &\perp\!\!\!\perp Z_2 \mid (W, Z_1) \\ Y &\perp\!\!\!\perp Z_1 \mid (X, W, Z_2) \quad (\text{where } W \text{ can be empty}) \end{aligned}$$

then $\{W, Z_2\}$ is better than $\{W, Z_1\}$, in that the expression using $\{W, Z_2\}$ has a lower asymptotic variance.

If we let $W = \phi$, $Z_1 = Q(X)$ and $Z_2 = Z$, these conditional independence statements become:

$$\begin{aligned} X &\perp\!\!\!\perp Z \mid Q(X) \\ Y &\perp\!\!\!\perp Q(X) \mid (X, Z) \end{aligned}$$

which are the conditional independence versions of Pearl’s Back Door conditions as given in section 5.5.4 of this thesis. This suggests that choosing a blocking set not equal to $Q(X)$ (if we can) is always more efficient than choosing $Z = Q(X)$.

Pursuing the ideas in [28] further, and applying them to CEGs, it seems highly probable that blocking sets produced from cuts further downstream a CEG from our intervention are better (more efficient) than sets produced by cuts close to the intervention. At the time of doing the initial research I ran a simulation for this problem (using binary variables as opposed to multivariate Normal), which again suggested that this result is true, all runs giving a lower variance to the estimate produced using a blocking set further downstream.

Again, this is a task that could be tackled using a BN, but it would be very easy to investigate using a CEG. We may well be able to produce CEG-based expressions requiring only the measurement / observation of certain functions of the problem variables – such expressions may be simpler, less costly, or allow us to produce estimates that would be impossible using a BN. The CEG would also allow us to compare the relative efficiency of different Back Door partitions using our CEG-based Back Door Theorem, for asymmetric manipulations of asymmetric problems.

6.3 Equivalence classes for CEGs

Different BNs can express the same conditional independence structure, so for example the first two DAGs in Figure 1 exhibit the property that $X \perp\!\!\!\perp Y \mid Z$, whereas the third does not.

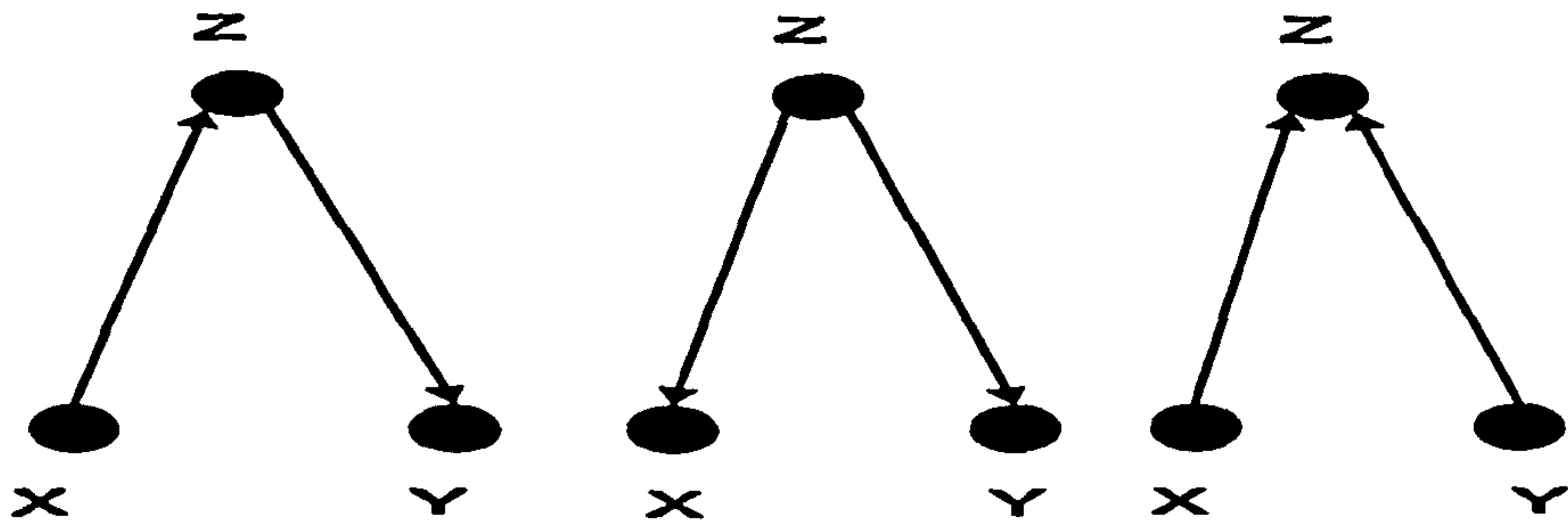


Figure 1: Three simple DAGs

This idea has been investigated by, among others, Andersson, Madigan and Perlman in [1]. They say that the first two DAGs in Figure 1 are *Markov equivalent*, and assign them to the same equivalence class.

Similarly, different CEGs can express the same conditional independence structure, and moreover if we allow reordering of a CEG, then there are circumstances where different CEGs correspond to exactly the same model – if the CEG is expressible as a BN, this BN can be drawn as a CEG in more than one way. Being able to determine the equivalence

classes of CEGs will be important from a theoretical perspective, but will also allow us to simplify elicitation in practical problems, and to design more efficient propagation algorithms. It should also help in the derivation of a d-separation theorem.

A couple of years ago I looked briefly at establishing equivalence classes for CEGs that were expressible as BNs: If we consider a problem with three variables A, B, C , that can be expressed graphically, then there are 25 possible DAGs that could be used to describe this problem. If we remove the vertex labels from these DAGs we are left with 6 distinct DAG-structures (given in Figure 2).

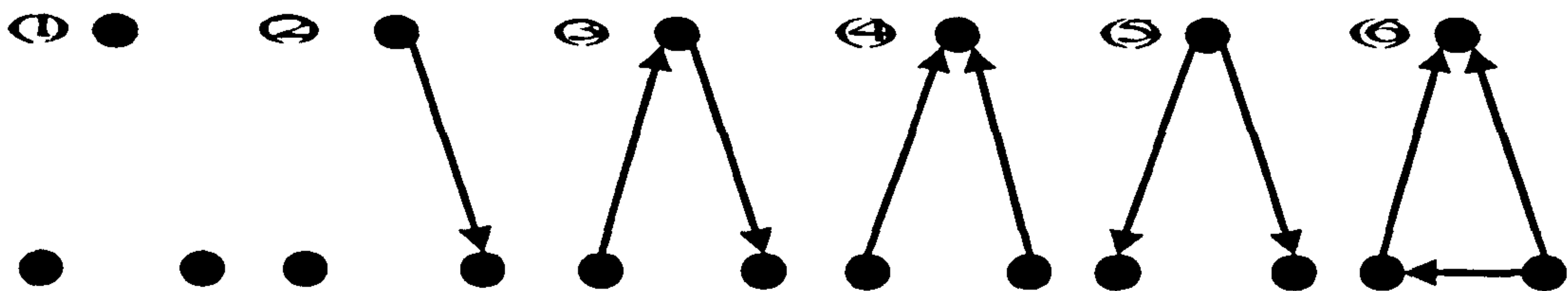


Figure 2: Six DAG-structures

If we let Ω_3 be the set of 25 DAGs, then applying the permutation group S_3 to Ω_3 splits it into 6 equivalence classes corresponding to the 6 distinct DAG-structures. Note that these are not the Markov-equivalence classes produced by Andersson, Madigan and Perlman, since here the three DAGs in Figure 1 would all be in different equivalence classes.

Now if we draw CEGs corresponding to the 25 DAGs, where each of A, B, C are binary, and remove all edge-labels and colours from these CEGs, we find that there are 8 distinct CEG-structures. Each DAG-structure in Figure 2 corresponds to one CEG-structure, except DAG-structure (2) which corresponds to three distinct CEG-structures, dependent on whether the independent variable (disconnected vertex) comes first, second or third in the CEG ordering.

Note that $8 = 2^3$ is the number of labelled undirected graphs with 3 vertices. This may appear to be coincidence, but if the vertices of these graphs are labelled A, B, C , and we add arrows to each edge so that the graphs are now directed and are compatible with the order ABC , then we get 8 distinct DAGs, corresponding to the graphs (1) to (6) in Figure 2, except that there are 3 variants of graph (2), with A, B or C as the disconnected vertex.

Now if we treat the graph labelled $A \rightarrow C \leftarrow B$ as distinct from the graph labelled $B \rightarrow C \leftarrow A$ (labellings of graph (4) in Figure 2) etc., then the DAG-structures (1) to (6)

in Figure 2 can be labelled in $6 + 3 + 1 + 2 + 2 + 1 = 15$ *ABC*-compatible ways. Each of these corresponds to an *ABC*-compatible CEG, so there are 15 *ABC*-compatible CEGs employing 8 CEG-structures.

But the numbers 6, 3, 1, 2, 2, 1 can be deduced without inspecting the DAG-structures in Figure 2!

DAG-structure (1) corresponds to 1 DAG – it is in an equivalence class with 1 member. It also corresponds to 1 CEG-structure, and the number of *ABC*-compatible labellings of the DAG is $|S_3| \div 1 = 6$. There are therefore 6 distinct CEGs (with different orderings) corresponding to DAG-structure 1, each using the same CEG-structure.

DAG-structure (2) corresponds to 6 DAGs – it is in an equivalence class with 6 members. It also corresponds to 3 CEG-structures, each with $|S_3| \div 6 = 1$ possible *ABC*-compatible ordering.

DAG-structure (3) corresponds to 6 DAGs and 1 CEG-structure. There is $|S_3| \div 6 = 1$ possible *ABC*-compatible ordering of this CEG-structure.

DAG-structure (4) corresponds to 3 DAGs and 1 CEG-structure. There are $|S_3| \div 3 = 2$ possible *ABC*-compatible orderings of this CEG-structure.

DAG-structure (5) corresponds to 3 DAGs and 1 CEG-structure. There are $|S_3| \div 3 = 2$ possible *ABC*-compatible orderings of this CEG-structure.

DAG-structure (6) corresponds to 6 DAGs and 1 CEG-structure. There is $|S_3| \div 6 = 1$ possible *ABC*-compatible ordering of this CEG-structure.

If we consider 4-variable problems, then there are 543 possible DAGs [1]. If we remove the vertex labels from these DAGs we are left with 31 distinct DAG-structures. I suspect that if we apply the permutation group S_4 to this set of 543 DAGs (Ω_4), then it will split into 31 equivalence classes corresponding to the 31 distinct DAG-structures. There are $64 = 2^6$ of these DAGs which are compatible with the ordering *ABCD*. I suggest that there are 64 possible CEG-structures. Note that 2^6 is the number of labelled undirected graphs with 4 vertices. Using the same principle as described above for the 3-variable case, we can show that the 31 DAG-structures can be labelled in 123 *ABCD*-compatible ways, suggesting that there are 123 *ABCD*-compatible CEGs, employing 64 CEG-structures.

For an n -variable symmetric problem with binary variables, I suggest there are $2^{1/2n(n-1)}$ possible CEG-structures. There may also be a single elegant formula for the number of *ABC...-compatible* CEGs, but in the short time I have spent looking at this I haven't found it.

Unfortunately, it is unlikely that this approach can be generalised for use with CEGs that are not expressible as BNs. Adding even a tiny degree of asymmetry to a problem prevents our using the DAG-structures described here, and without them we cannot use the permutation groups $\{S_n\}$, which give us the number and sizes of the equivalence classes. A possible way forward is suggested in section 6.4.

6.4 Further work

There are a number of ideas to do with the topology of the CEG which need to be investigated. These include allowing CEGs to have edges with zero probabilities (mentioned in sections 2.3 and 5.7); allowing different orders on different $w_0 \rightarrow w_\infty$ paths (mentioned in section 2.4); and considering CEGs where two positions on the same $w_0 \rightarrow w_\infty$ path can be stage-equivalent, or even equivalent (which could only happen in CEGs of infinite length). Beyond these there are seven principal areas in which we intend to do further research:

- (i) *the discovery and characterisation of equivalence classes for CEGs* – this was discussed briefly in section 6.3. We anticipate that there will be a number of *local* adjustments, swapping of positions and their outgoing edges, that can be made to any CEG, and the equivalence class of this CEG will be the set of CEGs that are closed under these local adjustments.
- (ii) *the production of a d-separation theorem for CEGs*. Progress has been made in this area (see chapter 3 and section 6.1), but we are really looking to find a condition analogous to that for BNs [41][30], whereby we can ask whether two sets of variables (or an event and a set of variables, or two events) are independent given an event (or possibly a set of variables), rather than just being able to say that given this event, this is independent of that.
- (iii) *the identification of appropriate applications*. I have indicated some fields in which asymmetric processes are common. To facilitate the testing of the theory and assess efficiency and applicability, we need to model problems from these fields and others. This will involve the model-representation and analysis of large sets of real data.
- (iv) *the development of Learning algorithms for CEGs*. J.Q. Smith in [45] has noted that the CEG gives a graphical coding of a factorisation of a likelihood when complete samples are observed, and suggests that this should (as with BNs) allow conjugate analyses to be performed. We expect that this will allow us to develop efficient Learning algorithms for CEGs.

- (v) *the further development of Propagation algorithms for CEGs*, including the development of the dynamic time-sliced CEG – this is discussed in sections 4.10 and 6.1.
- (vi) *the further analysis using CEGs of causal manipulated systems* – this is discussed in sections 5.7 and 6.1.
- (vii) *the use of CEGs for Decision Analysis*. As noted in section 1.2, there are close links between Influence diagrams and BNs, Decision Trees and Event Trees. In section 5.7 I suggested thinking of a Decision Tree as an Event Tree where most paths are subject to a number of interventions. If we extend this idea to CEGs we would get a Decision CEG, whose properties might derive from both our *Event CEG* and from Decision Trees.

I believe that the work presented in this thesis and in [53][45][61][60][62][63] has laid a solid foundation on which we can build. We have several papers on CEG-propagation in the pipeline, and I expect that we will make considerable progress in the areas listed above over the next couple of years.

References

- [1] S. A. Andersson, D. Madigan, and M. D. Perlman. A characterisation of Markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505–541, 1997.
- [2] T. Bedford and R. Cooke. *Probabilistic Risk Analysis: Foundations and Methods*, pages 99–151. Cambridge, 2001.
- [3] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian Networks. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, pages 115–123, Portland, Oregon, 1996.
- [4] H. J. Call and W. A. Miller. A comparison of approaches and implementations for automating Decision analysis. *Reliability Engineering and System Safety*, 30:115–162, 1990.
- [5] A. Cano, S. Moral, and A. Salmeron. Penniless Propagation in Join Trees. *International Journal of Intelligent Systems*, 15(11):1027–1059, 2000.
- [6] A. Cano, S. Moral, and A. Salmeron. Lazy evaluation in Penniless Propagation over Join Trees. *Networks*, 39(4):175–185, 2002.
- [7] G. A. Churchill. Accurate restoration of DNA sequences. In C. Gatsaris et al., editors, *Case Studies in Bayesian Statistics*, volume 2, pages 90–148. Springer-Verlag, 1995.
- [8] D. I. W. Coggan and C. N. Martyn. Time and chance: the stochastic nature of disease causation. *Lancet*, 365:1434–37, 2005.
- [9] Z. Covaliu and R. M. Oliver. Representation and solution of decision problems using sequential decision diagrams. *Management Science*, 41(12), 1995.
- [10] A. P. Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society, Series B*, 41:1–31, 1979.
- [11] A. P. Dawid. Causal inference without counterfactuals. *Journal of the American Statistical Association*, 95:407–448, 2000.
- [12] A. P. Dawid. Influence diagrams for causal modelling and inference. *International Statistical Review*, 70:161–89, 2002.
- [13] R. Forsyth. *Expert Systems: Principles and Case Studies*. Chapman and Hall, 1984.
- [14] S. French, editor. *Readings in Decision Analysis*. Chapman and Hall / CRC, 1989.
- [15] S. French and D. R. Insua. *Statistical Decision Theory*. Arnold, 2000.

- [16] N. Friedman and M. Goldszmidt. Learning Bayesian Networks with local structure. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 421–460. MIT Press, 1998.
- [17] C. W. J. Granger. Investigating causal relations by econometric models and cross spectral methods. *Econometrica*, 37:424–38, 1969.
- [18] D. Heckerman, D. Geiger, and D. Chickering. Learning Bayesian Networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.
- [19] D. Heckerman and R. Shachter. Decision-theoretic foundations for causal reasoning. *Journal of Artificial Intelligence Research*, 3:405–430, 1995.
- [20] D. Hume. A treatise of Human nature, 1739.
- [21] D. Hume. An enquiry concerning Human understanding, 1748.
- [22] C. Huygens. *Oeuvres completes*, volume 14. Martinus Nijhoff, 1920.
- [23] M. Jaeger. Probabilistic Decision Graphs: Combining verification and AI techniques for probabilistic inference. In *Proceedings of the 1st European Workshop on Probabilistic Graphical Models*, pages 81–88, 2002.
- [24] M. Jaeger, J. D. Nielsen, and T. Silander. Learning Probabilistic Decision Graphs. In *Proceedings of the 2nd European Workshop on Probabilistic Graphical Models*, pages 113–120, Leiden, 2004.
- [25] F. V. Jensen. *An introduction to Bayesian Networks*. UCL Press, 1996.
- [26] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer, second edition, 2007.
- [27] U. Kjaerulff. dHugin: a computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11:89–111, 1995.
- [28] M. Kuroki and M. Miyakawa. Covariate selection for estimating the causal effect of control plans by using causal diagrams. *Journal of the Royal Statistical Society Series B*, 65:209–222, 2003.
- [29] S. L. Lauritzen. *Graphical Models*. Oxford, 1996.
- [30] S. L. Lauritzen. Causal inference from graphical models. In O. E. Barndorff-Nielsen et al., editors, *Complex Stochastic Systems*. Chapman and Hall, 2001.

- [31] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H. G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.
- [32] S. L. Lauritzen and T. S. Richardson. Chain graph models and their causal interpretation. *Journal of the Royal Statistical Society Series B*, 64:321–361, 2002.
- [33] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to Expert Systems. *Journal of the Royal Statistical Society Series B*, 50(2):157–224, 1988.
- [34] S. L. Lauritzen and N. Wermuth. Graphical Models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.
- [35] A. L. Madsen and F. V. Jensen. LAZY propagation: A junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*, 113:263–245, 1999.
- [36] I. Martinez, C. Rodriguez, and A. Salmeron. Dynamic importance sampling in Bayesian Networks using factorisation of probability trees. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, pages 187–194, Prague, 2006.
- [37] R. M. Oliver and J. Q. Smith, editors. *Influence Diagrams, belief nets and Decision Analysis*. Wiley, 1990.
- [38] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings, Cognitive Science Society*, pages 329–334, 1985.
- [39] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.
- [40] J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82:669–710, 1995.
- [41] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge, 2000.
- [42] J. Pearl. Statistics and causal inference: A review. *Sociedad de Estadística e Investigación Operativa. Test*, 12(2):281–345, 2003.
- [43] R. O. Puch and J. Q. Smith. FINDS: A Training package to assess Forensic fibre evidence. In *Advances in Artificial Intelligence*, pages 420–429. Springer, 2004.
- [44] R. O. Puch, J. Q. Smith, and C. Bielza. Hierarchical Junction Trees: Conditional independence preservation and forecasting in Dynamic Bayesian Networks with het-

- erogeneous evolution. In J. A. Garnez et al., editors, *Advances in Bayesian Networks*. Springer Verlag, 2004.
- [45] E. M. Riccomagno and J. Q. Smith. The causal manipulation and Bayesian estimation of Chain Event Graphs. Research Report 05-16, CRiSM, 2005.
 - [46] J. M. Robins. A new approach to causal inference in mortality studies with sustained exposure periods – applications to control of the healthy worker survivor effect. *Mathematical Modelling*, 7:1393–1512, 1986.
 - [47] D. B. Rubin. Bayesian inference for causal effects: The role of randomization. *Annals of Statistics*, 6:34–58, 1978.
 - [48] L. J. Savage. *The Foundations of Statistics*. Wiley, 1954.
 - [49] G. Shafer. *The Art of Causal Conjecture*. MIT Press, 1996.
 - [50] P. P. Shenoy, G. Shafer, and K. Mellouli. Propagation of Belief Functions: A distributed approach. In *Proceedings of the 2nd Conference on Uncertainty in Artificial Intelligence*, Philadelphia, 1986.
 - [51] J. Q. Smith. *Decision Analysis: A Bayesian Approach*. Chapman and Hall, 1988.
 - [52] J. Q. Smith. Influence diagrams for statistical modelling. *Annals of Statistics*, 17:654–672, 1989.
 - [53] J. Q. Smith and P. E. Anderson. Conditional independence and Chain Event Graphs. *Artificial Intelligence*, 172:42–68, 2008.
 - [54] J. Q. Smith and K. N. Papamichail. Fast Bayes and the Dynamic Junction Forest. *Artificial Intelligence*, 107:99–124, 1999.
 - [55] J. Q. Smith and P. A. Thwaites. Decision trees. In E. L. Melnick and B. S. Everitt, editors, *Encyclopedia of Quantative Risk Analysis and Assessment*. Wiley, 2008.
 - [56] D. J. Spiegelhalter, A. P. Dawid, S. L. Lauritzen, and R. G. Cowell. Bayesian analysis in Expert Systems. *Statistical Science*, 8(3):219–247, 1993.
 - [57] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Springer-Verlag, 1993.
 - [58] M. Studeny. *Probabilistic Conditional Independence Structures*. Springer, 2005.
 - [59] M. Studeny and R. R. Bouckaert. On Chain Graph Models for description of conditional independence structures. *Annals of Statistics*, 26:1434–1495, 1998.

- [60] P. A. Thwaites and J. Q. Smith. Evaluating causal effects using Chain Event Graphs. In *Proceedings of the 3rd European Workshop on Probabilistic Graphical Models*, pages 291–300, Prague, 2006.
- [61] P. A. Thwaites and J. Q. Smith. Non-symmetric models, Chain Event Graphs and Propagation. In *Proceedings of the 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 2339–2347, Paris, 2006.
- [62] P. A. Thwaites and J. Q. Smith. Propagation on Chain Event Graphs. Research Report 447, University of Warwick Statistics Department, 2006.
- [63] P. A. Thwaites, J. Q. Smith, and R. G. Cowell. Propagation using Chain Event Graphs. Research Report 08-06, CRiSM, 2008.
- [64] N. Wermuth and S. L. Lauritzen. Graphical and recursive models for contingency tables. *Biometrika*, 70:537–552, 1983.